

Appendix A

MDA Software Applications Summary Table

Wireless Medical Enterprise Working Group

<http://www.tatrc.org>

Telemedicine & Advanced Technology Research Center (TATRC)

<http://www.tatrc.org>

US Army Medical Research & Materiel Command (USAMRMC Fort Detrick, MD)

Send information on additional applications to: TATRC WME

*Note: Last edit by DM 1/22/02 1330 (edit each rev)

| Application Name/Title | Type of Application | Description | Palm OS | WinCE/PC |
|------------------------|---------------------|-------------|---------|----------|
| | | | | |

| | | | | | | |
|---|--|---------------------------|--|---|-----|-----|
| | | | | | | |
| 4T Medical 1.3 | | Patient Charting software | | Database application for tracking personal medical information | Yes | NA |
| ABGPro | | Medical Calculator | | Arterial Blood Gas Interpretation | Yes | NA |
| Adobe Acrobat Reader for the Palm/Pocket PC | | Adobe Acrobat reader | | view Adobe Acrobat files | Yes | NA |
| Antibiotic Guide | | Medical Reference | | Antibiotic Guide from Johns Hopkins University | Yes | Yes |
| ApointisDoc Reader | | doc. Reader | | Doc reader | Yes | NA |
| Body Mass Index Calculator | | Medical Reference | | Calculates BMI using English Metric System | NA | NA |
| Cardio Textbook | | Medical Reference | | Cardiac problem reference (iSilo Format) | Yes | NA |
| CardioFILE 1.0.0 | | Patient Charting software | | For a Cardiologist to keep track of patients, Clinical Files, Progress Notes and Appointments | Yes | NA |
| Complete Surgical Knots | | Medical Reference | | What suture and what type of sutures to use on a wound | Yes | NA |
| Creatinine Clearance Calc 1.3 | | Medical Calc | | Calculate Creatinine clearance. How good their kidneys work | Yes | NA |
| CSpotRun | | doc. Reader | | Doc reader | Yes | NA |

| | | | | |
|--|-----------------------------|--|-----|-----|
| Current Medical Therapeutics 10 | Medical Reference | Up to date guidelines and recommendations for treatment for common acute medical conditions | Yes | NA |
| Deployment Vaccines and Medications | Medical Reference | From the CHPPM Website | Yes | NA |
| Digital Assist | Patient Charting software | Patient Charting Software | Yes | NA |
| DocuCare | Patient Charting software | Patient Charting Software | Yes | NA |
| Electrolytes First | Medical Reference | Chapters of metabolism: electrolytes hypo and hyper .kaliemia, nauremia, etc | Yes | NA |
| ePhysician | Medical Reference | Drug reference, Monitoring and absorption information, Lab test interferences | Yes | NA |
| ePocrates | Pharmacy, Medical Reference | Provides up to date information on drugs and drug interactions, dosage levels, Pregnancy/Lactation | Yes | NA |
| Ferri's Practical Guide to the Care of the Medical Patient | Medical Reference | A clear and concise reference for the busy clinician, that provides the very latest in clinical information, drug therapeutics and laboratory medicine | Yes | Yes |
| Gastroenterology | Medical Reference | for rotations in surgery or in internal medicine | Yes | NA |
| Global Expeditionary Medical System (GEMS) | Patient Charting software | Medical Surveillance System | Yes | Yes |
| Griffith's 5-Minute Clinical Consult | Medical Reference | An index that is organized by Diagnosis, Treatment, Medications and Follow-ups | Yes | Yes |
| HandBase | Relational Database | Relational Database | Yes | Yes |

| | | | | |
|-----------------------------------|---------------------------|---|-----|-----|
| MedCalc 3.2 | Medical Calc | Medical Calculator | Yes | NA |
| MedRules | Medical Calculator | Collection of common calculations/prediction tools in medicine | Yes | NA |
| MegaDoc | doc Reader | Doc reader | Yes | NA |
| Mobile Linkdoc | doc. Reader | Doc reader | Yes | NA |
| MobilePractice | Medical Reference | Provides patient management, medical reference, and current medical literature | Yes | NA |
| NAEPP Asthma Treatment Guidelines | Medical Reference | Provides succinct point-of-care asthma guidelines reference | Yes | NA |
| OEI Obesity Treatment Guidelines | Medical Reference | An interactive decision-support tool for implementing the OEI Guidelines | NA | NA |
| Ophthalmology Notes 1.11 | Medical Reference | Complete Ophthalmology Notes with different sources | Yes | NA |
| Orthopedic Surgery Notes 1.12 | Medical Reference | Complete ortho notes with pix included (iSilo Format) | Yes | NA |
| PanaRead | doc. Reader | Doc reader | Yes | NA |
| ParkStone | Medical Reference | medication management, referral generation on a network of palm-sized PCs (PPC) | NA | Yes |
| Patient Keeper | Patient Charting software | Patient Charting Software | Yes | NA |
| Patient Tracker 5.0 | Patient Charting software | Patient Charting Software | Yes | Yes |

| | | | | |
|--------------------------|----------------------------------|--|-----|----|
| PDA Ortho Bundle 1.0 | Medical Reference | Combines 3 Ortho documents(Ortho surgery, Ophthalmology, Surgical knots) | Yes | N/ |
| Peanut Reader | doc. Reader | Doc reader | Yes | N/ |
| Peds Doser | Medical Calculator | Pediatric medication calculator | Yes | N/ |
| PICU 3.0 | Medical Calc | calculates various drugs that are most commonly used in pediatric I.C.U | Yes | N/ |
| PocketChart | Patient Charting software | Patient Charting Software | NA | Yc |
| PocketPhoto | Support | Everything you need to put pictures on your Palm | Yes | N/ |
| PocketScript | Pharmacy, Medical Reference | obtain patient information, insurance coverage information, formulary listings and potential drug interactions | NA | N/ |
| Preg Dates | Medical Calculator | Pregnancy Wheel for determination of EDC | Yes | N/ |
| PregTrak 3.1 | Medical Calc | Program for tracking all your OB patients. | Yes | N/ |
| QED | doc. Reader | Doc reader | Yes | N/ |
| QuickOffice | Microsoft Office reader | Work with you Microsoft Office documents whenever you need to on your Palm! | Yes | Yc |
| QVADJS Express Reader GT | doc. Reader | Doc reader | Yes | N/ |
| Raphael 99 | Patient Charting software | Patient Charting Software | Yes | Yc |
| ReadyScript | Office Management/Coding/billing | real-time prescribing and patient management functions | NA | Yc |
| Redi-Reference Inc. | Pharmacy, Medical Reference | Provides handbooks in cardiology, obstetrics and clinical guidelines | Yes | N/ |
| Rich Reader | doc. Reader | Doc reader | Yes | N/ |
| Shots 2.2 | Medical Reference | Harmonized Immunization schedule | Yes | N/ |
| SmartDoc | doc. Reader | Doc reader | Yes | N/ |

| Tarascon Pharmacopeia | Pharmacy, Medical Reference | Provides up to date information on drugs and drug interactions, dosage levels, Pregnancy/Lactation | Yes | N/ |
|--|-----------------------------|--|-----|----|
| TealDoc | doc. Reader | Doc reader | Yes | N/ |
| TestTrakker | Diagnostic Tracker | TestTrakker™ for the Palm OS®, gives healthcare providers a simple, rapid, and intuitive risk management tool to assist them in tracking the countless numbers of laboratory and imaging tests that they order for their patients on a daily basis | Yes | N/ |
| The 1998 Guidelines for the Treatment of Sexually Transmitted Diseases | Medical Reference | STD reference | Yes | Ye |
| The CDC "Yellow Book" for Traveler's Illnesses | Medical Reference | Travelers Illnesses reference | Yes | Ye |
| The Harriet Lane Handbook | Medical Reference | This authoritative reference has been in use over the last 50 years for fast, and accurate treatment of a wide variety of pediatric patients. | Yes | Ye |
| The Medical Algorithms Project | Medical Reference | Look-up table for healthcare providers | NA | NA |
| The Redi-Reference handbooks | Pharmacy, Medical Reference | Quarterly updates for one year for current content for the e-book | Yes | NA |
| Treatment of Biological Warfare Agent Casualties | Medical Reference | Treatment reference for Biological Warfare | Yes | NA |
| US Naval Hospital Flight Surgeons Handbook | Manual, Reference | Naval Aerospace Medicine and a guide for dealing with the other complex administrative procedures known as "the system " | Yes | NA |
| WardWatch | Patient Charting software | To aid medical staff involved in ward rounds | Yes | NA |
| Washington Manual of Medical Therapeutics | Medical Reference | Physician Resource textbook that has been put in palm format | Yes | NA |

| | | | | |
|-------------|----------------------------------|---|-----|-----|
| Wireless MD | Pharmacy, Medical Reference | prescription writing, charge capture, physician directory, lab results, patient record information, e-mail, fax capabilities, and text-to-text messaging. | NA | NA |
| ZapCode 2.3 | Office Management/Coding/billing | Determines the Proper HCFA Evaluation and Management Code | Yes | Yes |
| Zdoc | doc. Reader | Doc reader | Yes | NA |

SOFTWARE APPENDIX

Including
Part I, pages 1 to 62, and
Part II, pages 1 to 32.

```

1 #
2 # PIDS_globals.cpk
3 #
4 #####
5 #
6 # PIDS (tm)
7 # Pharmacy Intervention Documentation System
8 # Copyright © 2002 Integrated Software Systems, L.L.C.
9 #
10 #####
11 #
12 # Version: 1.0
13 # Date: 8/15/2001
14 # Author: Todd Viegut & Brad Tice
15 #
16 #
17 # Miscellaneous functions.
18 #
19 # GLOBAL variable definitions used throughout the PIDS application.
20 #
21 variables;
22 #
23 numeric nInterventionKeyId;
24 numeric fUpdateMode = false;
25
26 #
27 numeric nLabValuesCreated;
28
29 #
30 numeric nTotalFrames = 14; # ADJUST THIS IF ANOTHER FRAME IS ADDED TO THE ARRAY BELOW
31 !!!
32 object objPIDSFrames[nTotalFrames] = PIDS_intro_frame, intervention_start_info_frame,
33 drug_related_problem_frame, prescription_type_frame, patient_risk_frame,
34 problem_type_frame, intervention_recommendation_frame, intervention_activity_frame,
35 intervention_result_frame, intervention_time_frame, patient_info_frame,
36 lab_values_frame, soap_follow_up_frame, save_intervention_frame;
37 numeric nContainsSelector[nTotalFrames] = false, false, true, true, true, true,
38 true, true, true, false, false, false, false;
39 object objInterventionSelectors[nTotalFrames] = drug_related_problem_selector,
40 drug_related_problem_selector, drug_related_problem_selector, rx_selector,
41 patient_risk_selector, problem_type_selector, intervention_recommendation_selector,
42 intervention_activity_selector, intervention_result_selector,
43 intervention_time_selector, intervention_time_selector, intervention_time_selector,
44 intervention_time_selector, intervention_time_selector;
45 numeric DRP_FRAME_NUMBER = 2;
46
47 # drug related problem arrays...
48 numeric nDRPSelected = 0;
49 numeric nTotalDRPFrames = 7; # ADJUST THIS IF ANOTHER FRAME IS ADDED TO THE ARRAY
50 BELOW !!!
51 object objDRPFrames[nTotalDRPFrames] = unnecessary_drug_therapy_frame,
52 wrong_drug_frame, dosage_too_low_frame, adverse_drug_reaction_frame,
53 dosage_too_high_frame, inappropriate_compliance_frame, need_therapy_frame;
54 object objDRPSelectors[nTotalDRPFrames] = unnecessary_drug_therapy_selector,
55 wrong_drug_selector, dosage_too_low_selector, adverse_drug_reaction_selector,
56 dosage_too_high_selector, inappropriate_compliance_selector, need_therapy_selector;
57 # NOTE here that element 0 is filler so that the array can be used as 1-based array...
58 numeric nDRPSelectionListOffsets[nTotalDRPFrames + 1] = 0, 0, 5, 10, 16, 22, 26, 32;
59
60 #
61 numeric nTotalTextEntries = 9; # ADJUST THIS IF ANOTHER BUTTON IS ADDED TO THE ARRAY
62 BELOW !!!
63 object objTextEntryButtons[nTotalTextEntries] = allergies_button, med_history_button,
64 family_history_button, social_history_button, subjective_button, objective_button,
65 assessment_button, plan_button, follow_up_button;
66 object objSearchTextEntryButtons[nTotalTextEntries] = search_allergies_button,
67 search_med_history_button, search_family_history_button, search_social_history_button,
68 search_subjective_button, search_objective_button, search_assessment_button,
69 search_plan_button, search_follow_up_button;
70 object objTextEntryTextFields[nTotalTextEntries] = txAllergies, txMedHistory,
71 txFamilyHistory, txSocialHistory, txSubjective, txObjective, txAssessment, txPlan,

```

```

1 txFollowUp;
2 string sTitles[nTotalTextEntries] = "Allergies", "Med History", "Family History",
3 "Social History", "Subjective", "Objective", "Assessment", "Plan", "Follow Up";
4 string sDBTextEntryMemos[nTotalTextEntries] = "", "", "", "", "", "", "", "", "", "";
5
6 numeric nAllergies = 0;
7 numeric nMedicalHistory = 1;
8 numeric nFamilyHistory = 2;
9 numeric nSocialHistory = 3;
10 numeric nSOAPSubjective = 4;
11 numeric nSOAPObjective = 5;
12 numeric nSOAPAssessment = 6;
13 numeric nSOAPPlan = 7;
14 numeric nSOAPFollowUp = 8;
15
16 #
17 # intervention weights...
18 # NOTE that all of their zero-th elements have a weight of 0 as filler so
19 # that it will be a 1-based array...
20 numeric nDRPWeights[35] = 0, 3, 3, 3, 4, 5, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3,
21 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3;
22 numeric nRXTypeWeights[3] = 0, 0, 0; # weight n/a to RX type so default to 0...
23 numeric nPatientRiskWeights[4] = 0, 3, 2, 1;
24 numeric nProblemTypeWeights[3] = 0, 2, 1;
25 numeric nRecommendationWeights[13] = 0, 3, 3, 3, 3, 3, 3, 3, 1, 1, 3, 2, 1;
26 numeric nActivityWeights[9] = 0, 3, 3, 3, 3, 2, 1, 4, 1;
27 numeric nResultWeights[5] = 0, 3, 2, 1, 1;
28 numeric nTimeWeights[5] = 0, 0, 0, 0, 0; # n/a
29 numeric nLabValueWeights[5] = 0, 0, 0, 0, 0;
30
31 # list box selections
32 numeric nTotalSelections = 10;
33 #numeric nNONE1 = 0;
34 #numeric nNONE2 = 1;
35 numeric nDRPSelection = 2;
36 numeric nRXTypeSelection = 3;
37 numeric nPatientRiskSelection = 4;
38 numeric nProblemTypeSelection = 5;
39 numeric nRecommendationSelection = 6;
40 numeric nActivitySelection = 7;
41 numeric nResultSelection = 8;
42 numeric nTimeSelection = 9;
43 numeric nDBInterventionSelections[nTotalSelections] = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;
44
45 #
46 numeric nTotalSearchableRecords;
47 numeric nCurrentDisplayedRecord;
48
49 #
50 numeric nCurrentFrame;
51
52 # preferences
53 numeric fSpeedEntryEnabled;
54 string sWirelessURL;
55 string sLocationID;
56 string sRotationID;
57 string sUserID;
58
59 end;
60
61
62

```

```
106 #
107 # PIDS_lists.cpk
108 #
109 #####
110 #
111 # PIDS (tm)
112 # Pharmacy Intervention Documentation System
113 # Copyright © 2002 Integrated Software Systems, L.L.C.
114 #
115 #####
116 #
117 # Version: 1.0
118 # Date: 1/6/2002
119 # Author: Todd Viegut & Brad Tice
120 #
121 #
122 # General purpose string arrays for PIDS (tm) pop-up lists.
123 #
124 variables;
125     numeric nAllergiesListSize = 23;
126     string allergies_list[nAllergiesListSize] = # text_entry_allergies_selector
127         "Other",
128         "Acetaminophen",
129         "Aspirin",
130         "Barbiturates",
131         "Benzodiazepines",
132         "Cephalosporins",
133         "Codeine",
134         "Digitalis",
135         "Erythromycin",
136         "Fish Product Derivatives",
137         "Iodine Derivatives",
138         "Local Anesthetics (Amides)",
139         "Local Anesthetics (Esters)",
140         "Nitrofurantoin",
141         "Non-Steroidal Anti-inflammatory Agents (Motrin)",
142         "Penicillins",
143         "Phenothiazines",
144         "Red Dye",
145         "Sulfonamide Derivatives (Sulfa)",
146         "Tartrazine (FD&C Yellow 5)",
147         "Tetracycline",
148         "Vaccines",
149         "No Known Allergies";
150     numeric nDrugsListSize = 207;
151     string drugs_list[nDrugsListSize] = # text_entry_med_history_selector
152         "Accolate",
153         "Accupril",
154         "Accutane",
155         "Aqualat",
156         "Adderall",
157         "Alesse",
158         "Allegra",
159         "Alphagen",
160         "Altace",
161         "Amaryl",
162         "Ambien",
163         "Amoxil",
164         "Aricept",
165         "Arthrotec",
166         "Atrovent",
167         "Augmentin",
168         "Avapro",
169         "Axid",
170         "Azmacort",
171         "Bactroban",
172         "Beconase AQ",
173         "Benzamycin",
174         "Blaxin",
175         "Blaxin Susp",
176         "Buspar",
```

177 "Buspar Dividose",
178 "Calan",
179 "Calan SR",
180 "Cardizem",
181 "Cardura",
182 "Ceftin",
183 "Cefzil",
184 "Celebra",
185 "Celebrex",
186 "Cipro",
187 "Claritin",
188 "Claritin Reditabs",
189 "Cleocin-T",
190 "Climara",
191 "Combivent",
192 "Coumadin",
193 "Covera-HS",
194 "Cozaar",
195 "Darvocet N-100",
196 "Daypro",
197 "Deltasone",
198 "Demdex",
199 "Demulen",
200 "Depakote",
201 "Desogen",
202 "Differin",
203 "Diflucan",
204 "Dilantin",
205 "Diovan",
206 "Dyazide",
207 "Effexor",
208 "Effexor",
209 "Effexor XR",
210 "Elocon",
211 "Estrace",
212 "Estraderm",
213 "Estratest",
214 "Estratest HS",
215 "Evista",
216 "Flonase",
217 "Flovent",
218 "Floxin",
219 "Fosamax",
220 "Glucophage",
221 "Glucotrol",
222 "Glynase",
223 "Humulin 7C/30",
224 "Humulin N",
225 "Humulin R",
226 "Hytrin",
227 "Hyzaar",
228 "Imdur",
229 "Imitrex",
230 "K-Dur",
231 "K-Dur",
232 "Klonopin",
233 "Lac-Hydrin",
234 "Lamisil",
235 "Lamisil Oral",
236 "Laroxin",
237 "Lasix",
238 "Lescol",
239 "Levaquin",
240 "Lipitor",
241 "Lo/Ovra",
242 "Lodine",
243 "LodineXL",
244 "Loestrin",
245 "Lopressor",
246 "Lorecig",
247 "Lotensin",

248 "Lotrel",
249 "Lotrisone",
250 "Macrobid",
251 "MetroGel Vaginal",
252 "Mevacor Risperdal",
253 "Miacalcin",
254 "Monopril",
255 "Nasacort",
256 "Nasacort AQ",
257 "Nascnex",
258 "Neurontin",
259 "Nitro-Dur",
260 "Nitrostat",
261 "Nizoral Shampoo",
262 "Nizoral tablets",
263 "Nizoral Topical",
264 "None",
265 "Nordette",
266 "Norvasc",
267 "Novolin 70/30",
268 "Novolin N",
269 "Novolin R",
270 "One Touch Test Strips",
271 "Ortho-Cept",
272 "Ortho-Cyclen",
273 "Ortho-Novum",
274 "Ortho Tri-Cyclen",
275 "Other",
276 "Oxycontin",
277 "Paxil",
278 "Pepcid",
279 "Pennergan",
280 "Plendil",
281 "Pravachol",
282 "Prednisone",
283 "Premarin",
284 "Premarin Vaginal",
285 "Premphase",
286 "Prempo",
287 "Prenate Ultra",
288 "Prevacid",
289 "Prilosec",
290 "Prinivil",
291 "Prinzide",
292 "Procardia",
293 "Propulsid",
294 "Proscar",
295 "Proventil",
296 "Proventil HFA",
297 "Provera",
298 "Prozac",
299 "Relafen",
300 "Remeron",
301 "Retin-A",
302 "Rezulin",
303 "Rhinocort",
304 "Ritalin",
305 "Serevent",
306 "Serzone",
307 "Singulair",
308 "Skelaxin",
309 "Suprax",
310 "Synthroid",
311 "Tapazole",
312 "Tegretol",
313 "Tenormin",
314 "Terazol 3",
315 "Terazol 7",
316 "Tiazac",
317 "Ticlid",
318 "Timoptic",

```
319      "Timoptic XE",
320      "Tobradex",
321      "Toprol XL",
322      "Tri-levlen",
323      "Triphasil",
324      "Trusopt",
325      "Tussionex",
326      "Ultram",
327      "Valium",
328      "Valtrex",
329      "Vancenase",
330      "Vanceril",
331      "Vaseretic",
332      "Vasotec",
333      "Ventolin",
334      "Verelan",
335      "Viagra",
336      "Vicodin",
337      "Vicodin ES",
338      "Vicoprofen",
339      "Vioxx",
340      "Voltaren XR",
341      "Wellbutrin",
342      "Wellbutrin",
343      "Westcort",
344      "Xalatan",
345      "Xanax",
346      "Zantac",
347      "Zaroxolyn",
348      "Zestoretic",
349      "Zestril",
350      "Ziac",
351      "Zithromax",
352      "Zocor",
353      "Zoloft",
354      "Zovirax Topical",
355      "Zyban",
356      "Zyprexa",
357      "Zyprexa",
358      "Zyrtec";
359  numeric nDiseasesListSize = 41;
360  string diseases_list[nDiseasesListSize] = # text_entry_family_history_selector
361      "None",
362      "Unknown",
363      "Acne or Psoriasis",
364      "AIDS",
365      "Allergies (Drug Related)",
366      "Allergies (Non-Drug Related)",
367      "Alzheimer's Disease",
368      "Angina",
369      "Anxiety Disorders",
370      "Asthma",
371      "Bipolar Disorders",
372      "Cancer",
373      "Congestive Heart Failure",
374      "Constipation & Diarrhea",
375      "Diabetes Mellitus",
376      "Epilepsy",
377      "GERD",
378      "Glaucoma",
379      "Gout",
380      "Headache Disorders",
381      "Heart Arrhythmias",
382      "Hormone Replacement Therapy",
383      "Hyperlipidemia",
384      "Hypertension",
385      "Infectious Disease",
386      "Inflammatory Bowel Disease",
387      "Multiple Sclerosis",
388      "Nausea & Vomiting",
389      "Nutrition Related Disorders",
```

```
390      "Osteoarthritis",
391      "Osteoporosis",
392      "Other",
393      "Pain Management",
394      "Pancreatitis",
395      "Peptic Ulcer Disease",
396      "Premenstrual Syndrome",
397      "Rheumatoid Arthritis",
398      "Sleep Disorders",
399      "Stroke",
400      "Substance-Related Disorders",
401      "Thyroid Disorders";
402
403      string social_list[5] = # text_entry_social_history_selector
404      "Caffeine",
405      "Diet and Exercise",
406      "EtOH",
407      "Tobacco",
408      "Illicit Drug Use";
409
410      #
411      # Lab Values lists...
412      #
413      string blood_pressure_list[3] =
414      "[Please select one]",
415      "Systolic Blood Pressure",
416      "Diastolic Blood Pressure";
417      string respiratory_value_list[4] =
418      "[Please select one]",
419      "Respiratory Rate (per minute)",
420      "PEFR (% personal best)",
421      "PEFR (% predicted)";
422      string blood_sugar_values[4] =
423      "[Please select one]",
424      "Fasting glucose (mg/dL)",
425      "Non-Fasting glucose (mg/dL)",
426      "Hemoglobin A1c (%)";
427      string lipid_values_list[6] =
428      "[Please select one]",
429      "Total Cholesterol",
430      "HDL",
431      "LDL",
432      "Triglycerides",
433      "TC/HDL ratio";
434      string osteoporosis_values_list[3] =
435      "[Please select one]",
436      "T-Score",
437      "Z-Score";
438      string liver_function_tests_list[4] =
439      "[Please select one]",
440      "ALT",
441      "AST",
442      "Alk Phos";
443      string electrolytes_list[5] =
444      "[Please select one]",
445      "Na+",
446      "K+",
447      "Cl-",
448      "PO4";
449      string anticoagulation_values_list[2] =
450      "[Please select one]",
451      "INR";
452      string renal_function_list[4] =
453      "[Please select one]",
454      "BUN",
455      "Cr",
456      "Creatinine Clearance (ClCr)";
457      string hematology_list[4] =
458      "[Please select one]",
459      "White Blood Cells",
460      "Segs",
```



```
461     "Bands";
462     #
463     # Lab values list...
464     string completed_lab_values[20] = "";
465 end;
466
467
```

```

468 #
469 # PIDS.csl
470 #
471 #####
472 #
473 # PIDS (tm)
474 # Pharmacy Intervention Documentation System
475 # Copyright © 2002 Integrated Software Systems, L.L.C.
476 #
477 #####
478 #
479 # Version: 1.0
480 # Date: 8/15/2001
481 # Author: Todd Viegut & Brad Tice
482 #
483 #
484 # MAIN SOURCE CODE FILE FOR PIDS (tm)
485 #
486 #
487 #
488 include "PIDS_date_lib.cpk";           # include intervention date routines
489 include "casler26.cpk";                 # include the CASL error definitions
490 include "casl_net_errors.cpk";          # include CASL network errors
491 include "PIDS_globals.cpk";             # GLOBAL variables
492 include "PIDS_lists.cpk";               # include the string lists for PIDS selectors
493 include "PIDS_sync.cpk";                # synchronization functions include file
494 include "PIDS_misc.cpk";                # miscellaneous function for PIDS
495 include "PIDS_wizards.cpk";
496 include "XML_parser.cpk";
497 #
498 #
499 #-----
500 # Standard functions for moving data between
501 # interventions and DB records and vice versa.
502 # Also included is the RESET method to clear
503 # all fields in order to apply another
504 # intervention.
505 #-----
506 #
507 function ResetFieldsForNextIntervention;
508     variables;
509     numeric nIndex;
510     end;
511     #
512     fUpdateMode = false;
513     #
514     nLabValuesCreated = 0;
515     nDRPSelected = 0;
516     nDBSynced = 0;
517     #
518     # Clear start fields
519     txLocationID.display = sLocationID;
520     #hide intervention_date_button;
521     intervention_date_button.display = GetCurrentInterventionDate;
522     txRotationID.display = sRotationID;
523     txUserID.display = sUserID;
524     txPrescribedDrug.display = "";
525     txDisease.display = "";
526     txPrescriber.display = "";
527     #
528     # Clear db numeric fields...
529     for nIndex = 0, nIndex < nTotalSelections;
530         nDBInterventionSelections[nIndex] = 0;
531     next nIndex + 1;
532     #
533     # Clear selectors...
534     for nIndex = 0, nIndex < nTotalFrames;
535         hide objInterventionSelectors[nIndex];
536         objInterventionSelectors[nIndex].selected = 0;
537     next nIndex + 1;
538     #

```

```

539     # Clear DRP selectors...
540     for nIndex = 0, nIndex < nTotalDRPFFrames;
541         hide objDRPSelectors[nIndex];
542         objDRPSelectors[nIndex].selected = 0;
543     next nIndex + 1;
544     #
545     txPatientID.display = "";
546     #
547     # Clear db memo fields
548     for nIndex = 0, nIndex < nTotalTextEntries;
549         sDBTextEntryMemos[nIndex] = "";
550     next nIndex + 1;
551     #
552     # Clear text fields... etc.
553     for nIndex = 0, nIndex < nTotalTextEntries;
554         objTextEntryTextFields[nIndex].display = "";
555     next nIndex + 1;
556     #
557     # Clear lab values list...
558     call ClearLabValuesList;
559     #
560     # Miscellaneous fields...
561     txtLabValues.display = "";
562 end;

563
564 function MoveInterventionToDBFields;
565     variables;
566         string sValue;
567         numeric nIndex;
568     end;
569
570     # set primary key index...
571     nDBPIDSKeyField = GetInterventionKeyId;
572     nDBSynced = 0;
573     #
574     sDBLocationID = txLocationID.display;
575     sDBInterventionDate = intervention_date_button.display;
576     sDBRotationID = txRotationID.display;
577     sDBUserID = txUserID.display;
578     sDBPrescribedDrug = txPrescribedDrug.display;
579     sDBDisease = txDisease.display;
580     sDBPrescriber = txPrescriber.display;
581     #
582     # set db numeric fields...
583     nDBDrugRelatedProblemField = nDBInterventionSelections[nDRPSelection];
584     nDBRXTypeField = nDBInterventionSelections[nRXTypeSelection];
585     nDBPatientRiskField = nDBInterventionSelections[nPatientRiskSelection];
586     nDBProblemTypeField = nDBInterventionSelections[nProblemTypeSelection];
587     nDBRecommendationField = nDBInterventionSelections[nRecommendationSelection];
588     nDBActivityField = nDBInterventionSelections[nActivitySelection];
589     nDBResultField = nDBInterventionSelections[nResultSelection];
590     nDBTimeField = nDBInterventionSelections[nTimeSelection];
591     #
592     #
593     for nIndex = 0, nIndex < nTotalTextEntries;
594         get objTextEntryTextFields[nIndex], sValue;
595         sDBTextEntryMemos[nIndex] = sValue;
596     next nIndex + 1;
597     #
598     # set db memo fields
599     sDBPatientIDField = txPatientID.display;
600     sDBAllergiesField = sDBTextEntryMemos[nAllergies];
601     sDBMedicalHistoryField = sDBTextEntryMemos[nMedicalHistory];
602     sDBFamilyHistoryField = sDBTextEntryMemos[nFamilyHistory];
603     sDBSocialHistoryField = sDBTextEntryMemos[nSocialHistory];
604     #
605     sDBSOAPSSubjectiveField = sDBTextEntryMemos[nSOAPSSubjective];
606     sDBSOAPObjectiveField = sDBTextEntryMemos[nSOAPObjective];
607     sDBSOAPAssessmentField = sDBTextEntryMemos[nSOAPAssessment];
608     sDBSOAPPlanField = sDBTextEntryMemos[nSOAPPlan];
609     sDBSOAPFollowUpField = sDBTextEntryMemos[nSOAPFollowUp];

```

```
610     #
611     sDBLabValues = txtLabValues.display;
612     nDBDOI_Score = CalculateDOI_Score;
613 end;
614
615 function MoveDBFieldsToIntervention;
616     variables;
617     numeric nIndex;
618 end;
619 #
620 # set primary key index...
621 nInterventionKeyId = nDBPIDSKeyField;
622 #
623 txLocationID.display = sDBLocationID;
624 hide intervention_date_button;
625 intervention_date_button.display = sDBInterventionDate;
626 txRotationID.display = sDBRotationID;
627 txUserID.display = sDBUserID;
628 txPrescribedDrug.display = sDBPrescribedDrug;
629 txDisease.display = sDBDisease;
630 txPrescriber.display = sDBPrescriber;
631 #
632 nDRPSelected = GetDRPTypeIndex(nDBDrugRelatedProblemField);
633 #
634 nIndex = 0;
635 #
636 for nIndex = 0, nIndex < nTotalDRPFrames;
637     if (nIndex = nDRPSelected);
638         objDRPSelectors[nIndex - 1].selected =
639             GetDRPTypeSubIndex(nDBDrugRelatedProblemField);
640     else;
641         objDRPSelectors[nIndex].selected = 0;
642     end_if;
643 next nIndex + 1;
644 #
645 # Set DB numeric fields...
646 nDBInterventionSelections[nDRPSelection] = nDBDrugRelatedProblemField;
647 nDBInterventionSelections[nRXTypeSelection] = nDBRXTypeField;
648 nDBInterventionSelections[nPatientRiskSelection] = nDBPatientRiskField;
649 nDBInterventionSelections[nProblemTypeSelection] = nDBProblemTypeField;
650 nDBInterventionSelections[nRecommendationSelection] = nDBRecommendationField;
651 nDBInterventionSelections[nActivitySelection] = nDBActivityField;
652 nDBInterventionSelections[nResultSelection] = nDBResultField;
653 nDBInterventionSelections[nTimeSelection] = nDBTimeField;
654 #
655 objInterventionSelectors[nDRPSelection].selected = nDRPSelected;
656 objInterventionSelectors[nRXTypeSelection].selected =
657     nDBInterventionSelections[nRXTypeSelection];
658 objInterventionSelectors[nPatientRiskSelection].selected =
659     nDBInterventionSelections[nPatientRiskSelection];
660 objInterventionSelectors[nProblemTypeSelection].selected =
661     nDBInterventionSelections[nProblemTypeSelection];
662 objInterventionSelectors[nRecommendationSelection].selected =
663     nDBInterventionSelections[nRecommendationSelection];
664 objInterventionSelectors[nActivitySelection].selected =
665     nDBInterventionSelections[nActivitySelection];
666 objInterventionSelectors[nResultSelection].selected =
667     nDBInterventionSelections[nResultSelection];
668 objInterventionSelectors[nTimeSelection].selected =
669     nDBInterventionSelections[nTimeSelection];
670 #
671 # set db memo fields
672 txPatientId.display = sDBPatientIDField;
673 sDBTextEntryMemos[nAllergies] = sDBAllergiesField;
674 sDBTextEntryMemos[nMedicalHistory] = sDBMedicalHistoryField;
675 sDBTextEntryMemos[nFamilyHistory] = sDBFamilyHistoryField;
676 sDBTextEntryMemos[nSocialHistory] = sDBSocialHistoryField;
677 #
678 sDBTextEntryMemos[nSOAPSSubjective] = sDBSOAPSSubjectiveField;
679 sDBTextEntryMemos[nSOAPObjective] = sDBSOAPObjectiveField;
680 sDBTextEntryMemos[nSOAPAssessment] = sDBSOAPAssessmentField;
```

```

673     sDBTextEntryMemos[nSOAPPlan] = sDBSOAPPlanField;
674     sDBTextEntryMemos[nSOAPFollowUp] = sDBSOAPFollowUpField;
675     #
676     nIndex = 0;
677     #
678     for nIndex = 0, nIndex < nTotalTextEntries;
679         objTextEntryTextFields[nIndex].display = sDBTextEntryMemos[nIndex];
680     next nIndex + 1;
681     #
682     txtLabValues.display = sDBLabValues;
683 end;
684
685 #
686 #-----
687 # MENU OBJECT DEFINITIONS
688 #-----
689 #
690 menu_top mtOptions;
691     display "Options";
692 end;
693
694 menu_item miSearch, mtOptions;
695     display "Search for intervention...";
696 end;
697
698 menu_item miSync, mtOptions;
699     display "Perform wireless sync...";
700 end;
701
702 #menu_item miPurgeDB, mtOptions;
703 #     display "Purge interventions...";
704 #end;
705
706 menu_top mtSetup;
707     display "Setup";
708 end;
709
710 menu_item miPreferences, mtSetup;
711     display "Set preferences...";
712 end;
713
714 #menu_item miDeletePIDSDatabase, mtSetup;
715 #     display "Delete database...";
716 #end;
717
718 function SelectInterventionDate;
719     sDBInterventionDate = GetCurrentInterventionDate;
720     intervention_date_button.display = sDBInterventionDate;
721 end;
722
723 function DrugRelatedProblemTypeSelected;
724     variables;
725         numeric nItemIndex;
726     end;
727     get drug_related_problem_selector, nItemIndex;
728     nDRPSelected = nItemIndex;
729     if (nItemIndex > 0) and (nItemIndex <= nTotalDRPFrames);
730         hide drug_related_problem_frame;
731         show objDRPSelectors[nItemIndex - 1]; # - 1 because of 0 based arrays...
732         show objDRPFrames[nItemIndex - 1]; # - 1 because of 0 based arrays...
733     end_if;
734 end;
735
736 function DRPPreviousForm;
737     hide objDRPFrames[nDRPSelected - 1]; # - 1 because of 0 based arrays...
738     hide objDRPSelectors[nDRPSelected - 1]; # - 1 because of 0 based arrays...
739     show drug_related_problem_frame;
740 end;
741
742 function DRPNextForm;
743     variables;

```

```

744     object objSelector;
745     numeric nDRPSelectedItemIndex;
746     #numeric nSelectedItemIndex;
747 end;
748 #
749 # Set the intervention selector DB values...
750 objSelector = objDRPSelectors[nDRPSelected - 1];
751 nDRPSelectedItemIndex = objSelector.selected;
752 if (nDRPSelectedItemIndex > 0);
753     nDBInterventionSelections[nCurrentFrame] = nDRPSelectionListOffsets[nDRPSelected]
754     + nDRPSelectedItemIndex;
755 #
756 # Drop the DRP chosen into the Assessment field of the SOAP screen...
757 if (objPIDSFrames[nCurrentFrame] = patient_info_frame) and
    (length(txAssessment.display) = 0);
758     txAssessment.display =
759     GetDrugRelatedProblemData(nDBInterventionSelections[nCurrentFrame]);
760 end_if;
761 #
762 # Now switch frames...
763 hide objDRPFrames[nDRPSelected - 1];    # - 1 because of 0 based arrays...
764 hide objDRPSelectors[nDRPSelected - 1]; # - 1 because of 0 based arrays...
765 nCurrentFrame = nCurrentFrame + 1;
766 call ShowFrame;
767 end_if;
768 end;
769
770 function PreviousForm;
771     if (nCurrentFrame > 0);
772         if (nCurrentFrame > 1);
773             call HideFrame;
774             nCurrentFrame = nCurrentFrame - 1;
775         end_if;
776         call ShowFrame;
777     end_if;
778 end;
779
780 function NextForm;
781     variables;
782     object objSelector;
783     numeric nSelectedItemIndex;
784 end;
785 if (nCurrentFrame < (nTotalFrames - 1));
786     if (nContainsSelector[nCurrentFrame]);
787         objSelector = objInterventionSelectors[nCurrentFrame];
788         nSelectedItemIndex = objSelector.selected;
789         if (nSelectedItemIndex > 0);
790             if (nCurrentFrame = DRP_FRAME_NUMBER);
791                 call DrugRelatedProblemTypeSelected;
792             else;
793                 #
794                 # Set the intervention selector DB values...
795                 nDBInterventionSelections[nCurrentFrame] = nSelectedItemIndex;
796                 #
797                 # Now switch frames...
798                 call HideFrame;
799                 nCurrentFrame = nCurrentFrame + 1;
800                 call ShowFrame;
801             end_if;
802         end_if;
803     else;
804         if (objPIDSFrames[nCurrentFrame] = patient_info_frame) and
805             (length(txSubjective.display) = 0);
806             txSubjective.display = GetPatientInfoData();
807         end_if;
808         #
809         # UNCOMMENT LATER!!!!
810         #
811         if (objPIDSFrames[nCurrentFrame] = patient_info_frame) and
812             (length(txAssessment.display) = 0);
813             txAssessment.display =

```

```

        GetDrugRelatedProblemData(nDBInterventionSelections[DRP_FRAME_NUMBER]);
810     end_if;
811     #
812     # Drop the DRP chosen into the Assessment field of the SOAP screen...
813     #
814     call HideFrame;
815     nCurrentFrame = nCurrentFrame + 1;
816     call ShowFrame;
817     end_if;
818 end_if;
819 end;
820
821 #
822 #-----
823 # INVOKER functions for the seven frames that the
824 # drug related problem frame directs to...
825 #-----
826 #
827
828 function ItemSelected;
829     variables;
830     object objSelector;
831     numeric nDRPSelectedItemIndex;
832     numeric nSelectedItemIndex;
833 end;
834 if fSpeedEntryEnabled;
835     if (nCurrentFrame < (nTotalFrames - 1));
836         objSelector = objInterventionSelectors[nCurrentFrame];
837         nSelectedItemIndex = objSelector.selected;
838         if (nSelectedItemIndex > 0);
839             if (nCurrentFrame = DRP_FRAME_NUMBER) and (invoker.object_string = "DRP");
840                 call DrugRelatedProblemTypeSelected;
841             else;
842                 #
843                 # Set the intervention selector DB values...
844                 if nCurrentFrame = DRP_FRAME_NUMBER;
845                     objSelector = objDRPSelectors[nDRPSelected - 1];
846                     nDRPSelectedItemIndex = objSelector.selected;
847                     nDBInterventionSelections[nCurrentFrame] =
848                         nDRPSelectionListOffsets[nDRPSelected] + nDRPSelectedItemIndex;
849                 else;
850                     nDBInterventionSelections[nCurrentFrame] = nSelectedItemIndex;
851                 end_if;
852                 #
853                 # Now switch frames...
854                 call HideFrame;
855                 nCurrentFrame = nCurrentFrame + 1;
856                 call ShowFrame;
857             end_if;
858         end_if;
859     end_if;
860 end;
861
862 #
863 # Search Screen Function Definitions:
864 #
865 function UpdateSearchDisplay(numeric nRecord);
866     if ((nTotalSearchableRecords = 0) or ((nRecord = 1) and (nTotalSearchableRecords =
867         1)));
868         previous_record_button.display = char(5);
869         next_record_button.display = char(6);
870     else_if (nRecord = 1);
871         previous_record_button.display = char(5);
872         next_record_button.display = char(3);
873     else_if (nRecord = nTotalSearchableRecords);
874         previous_record_button.display = char(2);
875         next_record_button.display = char(6);
876     else;
877         previous_record_button.display = char(2);
878         next_record_button.display = char(3);

```

```

878     end_if;
879     record_set_label.display = string(nRecord, "#") + " of " +
        string(nTotalSearchableRecords, "#");
880 end;
881
882 function ClearSearchDisplay;
883     variables;
884         numeric nIndex;
885     end;
886     #
887     nCurrentDisplayedRecord = 0;
888     nTotalSearchableRecords = 0;
889     UpdateSearchDisplay(nCurrentDisplayedRecord);
890     #
891     hide search_edit_button;
892     hide search_delete_intervention_button;
893     #
894     nIndex = 0;
895     #
896     # change all button bitmaps to "empty" state...
897     for nIndex = 0, nIndex < nTotalTextEntries;
898         objSearchTextEntryButtons[nIndex].display = char(11);
899     next nIndex + 1;
900     #
901     txSearchPatientID.display = "";
902     txSearchDate.display = "";
903     txSearchDRPDescription.display = "";
904     txSearchRxType.display = "";
905     txSearchPatientRisk.display = "";
906     txSearchProblemType.display = "";
907     txSearchPrescribedDrug.display = "";
908     txSearchRecommendation.display = "";
909     txSearchActivity.display = "";
910     txSearchResult.display = "";
911     txSearchTime.display = "";
912     txSearchDOIScore.display = "";
913 end;
914
915 function IsRecordSynced(numeric nRecord) as numeric;
916     if (dba_ffOpenDB(dbPidsXML, "dbPidsXML"));
917         if (dba_ffReadRecordFromRecNo(dbPidsXML, nRecord));
918             IsRecordSynced = nDBXMLSynced;
919         else;
920             IsRecordSynced = -1; # Indicated an error occurred...
921         end_if;
922     dba_CloseDB(dbPidsXML);
923 else;
924     IsRecordSynced = -1; # Indicated an error occurred...
925 end_if;
926 end;
927
928 function PopulateSearchScreen(numeric nRecord);
929     variables;
930         object oText;
931         string sCurrentChar;
932         string sValue;
933         numeric nIndex;
934         numeric nDRPSubIndex;
935         numeric nSynced;
936     end;
937     #
938     nSynced = IsRecordSynced(nRecord);
939     if (dba_ffOpenDB(dbPids, "dbPids"));
940         if (dba_ffReadRecordFromRecNo(dbPids, nRecord));
941             #
942             if (not (nSynced = 0));
943                 hide search_edit_button;
944             else;
945                 show search_edit_button;
946             end_if;
947             show search_delete_intervention_button;

```



```

948      #
949      nIndex = GetDRPTypeIndex(nDBDrugRelatedProblemField);
950      #
951      nDRPSubIndex = GetDRPTypeSubIndex(nDBDrugRelatedProblemField);
952      #
953      if nIndex = 1;
954          txSearchDRPDescription.display =
              unnecessary_drug_therapy_list[nDRPSubIndex];
955      else_if nIndex = 2;
956          txSearchDRPDescription.display = wrong_drug_list[nDRPSubIndex];
957      else_if nIndex = 3;
958          txSearchDRPDescription.display = dosage_too_low_list[nDRPSubIndex];
959      else_if nIndex = 4;
960          txSearchDRPDescription.display = adverse_drug_reaction_list[nDRPSubIndex];
961      else_if nIndex = 5;
962          txSearchDRPDescription.display = dosage_too_high_list[nDRPSubIndex];
963      else_if nIndex = 6;
964          txSearchDRPDescription.display =
              inappropriate_compliance_list[nDRPSubIndex];
965      else_if nIndex = 7;
966          txSearchDRPDescription.display = need_therapy_list[nDRPSubIndex];
967      end_if;
968      #
969      txSearchPatientID.display = sDBPatientIDField;
970      #
971      txSearchDate.display = sDBInterventionDate;
972      #
973      txSearchRxType.display = rx_type_list[nDBRXTypeField];
974      #
975      txSearchPatientRisk.display = patient_risk_type_list[nDBPatientRiskField];
976      #
977      txSearchProblemType.display = problem_type_list[nDBProblemTypeField];
978      #
979      txSearchPrescribedDrug.display = sDBPrescribedDrug;
980      #
981      txSearchRecommendation.display =
          intervention_recommendation_list[nDBRecommendationField];
982      #
983      txSearchActivity.display = intervention_activity_list[nDBActivityField];
984      #
985      txSearchResult.display = intervention_result_list[nDBResultField];
986      #
987      txSearchTime.display = intervention_time_list[nDBTimeField];
988      #
989      txSearchDOIScore.display = string(nDBDOIScore, "#");
990      #
991      search_allergies_button.object_string = sDBAllergiesField;
992      search_med_history_button.object_string = sDBMedicalHistoryField;
993      search_family_history_button.object_string = sDBFamilyHistoryField;
994      search_social_history_button.object_string = sDBSocialHistoryField;
995      #
996      search_subjective_button.object_string = sDBSOAPSubjectiveField;
997      search_objective_button.object_string = sDBSOAPObjectiveField;
998      search_assessment_button.object_string = sDBSOAPAssessmentField;
999      search_plan_button.object_string = sDBSOAPPlanField;
1000      search_follow_up_button.object_string = sDBSOAPFollowUpField;
1001
1002      #
1003      nIndex = 0;
1004      #
1005      for nIndex = 0, nIndex < nTotalTextEntries;
1006          oText = objSearchTextEntryButtons[nIndex];
1007          sValue = oText.object_string;
1008          if (length(sValue) > 0);
1009              get objSearchTextEntryButtons[nIndex], sCurrentChar;
1010              if (sCurrentChar <> char(9));
1011                  put objSearchTextEntryButtons[nIndex], char(9);
1012              end_if;
1013          else;
1014              get objSearchTextEntryButtons[nIndex], sCurrentChar;
1015              if (sCurrentChar <> char(11));

```

```

1016         put objSearchTextEntryButtons[nIndex], char(11);
1017     end_if;
1018     end_if;
1019     next nIndex + 1;
1020     # = nDBDOIScore;
1021 else;
1022     call ClearSearchDisplay;
1023     ErrorMsg("Unable to read intervention from the PIDS database.");
1024 end_if;
1025 dba_ClosedB(dbPids);
1026 else;
1027     call ClearSearchDisplay;
1028     ErrorMsg("Unable to open PIDS intervention database.");
1029 end_if;
1030 end;
1031
1032 function InitSearch();
1033     variables;
1034     numeric nIndex;
1035 end;
1036 #
1037 call ClearSearchDisplay;
1038 #
1039 if (dba_ffOpenDB(dbPids, "dbPids"));
1040     #
1041     # Retrieve the total records currently in the database...
1042     nIndex = 0;
1043     seek_start dbPids, 0;
1044     get_fields dbPids;
1045     #
1046     while (errorcode = ce_no_error);
1047         nIndex = nIndex + 1;
1048         get_fields dbPids;
1049     end_while;
1050     #
1051     if (errorcode <> ce_eof);
1052         ErrorMsg("Problem reading interventions from the PIDS database.");
1053     else;
1054         nTotalSearchableRecords = nIndex;
1055         if (nTotalSearchableRecords > 0);
1056             nCurrentDisplayedRecord = 1;
1057             call UpdateSearchDisplay(nCurrentDisplayedRecord);
1058             call PopulateSearchScreen(nCurrentDisplayedRecord - 1);
1059         end_if;
1060     end_if;
1061     dba_ClosedB(dbPids);
1062 else;
1063     ErrorMsg("Unable to open PIDS intervention database.");
1064 end_if;
1065 end;
1066 #
1067 #-----
1068 # Previous and next record functions for the
1069 # search screen...
1070 #-----
1071 #
1072 function PreviousRecord;
1073     if (nCurrentDisplayedRecord > 1);
1074         nCurrentDisplayedRecord = nCurrentDisplayedRecord - 1;
1075         call UpdateSearchDisplay(nCurrentDisplayedRecord);
1076         call PopulateSearchScreen(nCurrentDisplayedRecord - 1);
1077     end_if;
1078 end;
1079
1080 function NextRecord;
1081     if (nCurrentDisplayedRecord < nTotalSearchableRecords);
1082         nCurrentDisplayedRecord = nCurrentDisplayedRecord + 1;
1083         call UpdateSearchDisplay(nCurrentDisplayedRecord);
1084         call PopulateSearchScreen(nCurrentDisplayedRecord - 1);
1085     end_if;
1086 end;

```

```

1087
1088 function ShowTextEntryScreen;
1089     variables;
1090     string sValue;
1091     numeric nIndex;
1092     numeric nTextEntryIndex;
1093 end;
1094 #
1095 # string sTitles[nTotalTextEntries] = "Allergies", "Med History", "Family History",
1096 # "Social History",
1097 # "Subjective", "Objective", "Assessment", "Plan",
1098 # "Follow Up";
1099 # Only these 4 text entry screens have pop up capability...
1100 #
1101 for nIndex = 0, nIndex < nTotalTextEntries;
1102     if objTextEntryButtons[nIndex] = invoker;
1103         nTextEntryIndex = nIndex;
1104         #
1105         text_entry_frame.display = sTitles[nIndex];
1106         accept_text_entry_button.object_string = sTitles[nIndex];
1107         #
1108         generic_selector_button.object_string = sTitles[nIndex];
1109         generic_selector_label.display = sTitles[nIndex];
1110         #
1111         get objTextEntryTextFields[nIndex], sValue;
1112         txEntry.display = sValue;
1113     end_if;
1114 next nIndex + 1;
1115 #
1116 call HideFrame;
1117 show text_entry_frame;
1118 if ((nTextEntryIndex = 0) or
1119     (nTextEntryIndex = 1) or
1120     (nTextEntryIndex = 2) or
1121     (nTextEntryIndex = 3));
1122     show generic_selector_label;
1123     show generic_selector_button;
1124 else;
1125     hide generic_selector_label;
1126     hide generic_selector_button;
1127 end_if;
1128 end;
1129
1130 function AcceptTextEntry;
1131     variables;
1132     numeric nIndex;
1133 end;
1134 for nIndex = 0, nIndex < nTotalTextEntries;
1135     if sTitles[nIndex] = accept_text_entry_button.object_string;
1136         objTextEntryTextFields[nIndex].display = txEntry.display;
1137     end_if;
1138 next nIndex + 1;
1139 hide text_entry_frame;
1140 call ShowFrame;
1141 end;
1142
1143 function CancelTextEntry;
1144     hide text_entry_frame;
1145     call ShowFrame;
1146 end;
1147
1148 function miSync;
1149     variables;
1150     numeric selection;
1151 end;
1152 selection = message_box(1, "Perform Wireless Sync", "Would you like to perform a
1153 wireless sync via the web?", "Yes", "Cancel", "");
1154 if (selection = 0);
1155     call SyncAllInterventions(sWirelessURL);
1156 end_if;
1157 end;

```

```

1155
1156 function miPreferences;
1157     call HideFrame;
1158     hide intervention_search_frame;
1159     put txPrefsWirelessSyncURL, sWirelessURL;
1160     put txPrefsLocationID, sLocationID;
1161     put txPrefsRotationID, sRotationID;
1162     put txPrefsUserID, sUserID;
1163     put cbPrefsSpeedEntryCheckbox, fSpeedEntryEnabled;
1164     show pids_preferences_frame;
1165 end;
1166
1167 function miDeletePIDSDatabase;
1168     if (message_box(1,"Delete PIDS Database", "Are you sure you want to delete PIDS (all
1169         data will be lost)?", "Yes","Cancel","", "") = 0);
1170         dba_DeleteDB(dbPids, "dbPids");
1171     end_if;
1172     if (message_box(1,"Delete Preferences Database", "Are you sure you want to delete
1173         preferences (all data will be lost)?", "Yes","Cancel","", "") = 0);
1174         dba_DeleteDB(dbPidsPrefs, "dbPidsPrefs");
1175     end_if;
1176 end;
1177
1178 function miSearch;
1179     call HideFrame();
1180     hide pids_preferences_frame;
1181     show intervention_search_frame;
1182     call InitSearch();
1183 end;
1184
1185 #-----
1186 # Auxiliary invoker methods...
1187 #-----
1188 #
1189 function AcceptPreferences;
1190     variables;
1191     numeric nReturnCode;
1192     end;
1193     nReturnCode = SavePreferences;
1194     #
1195     hide pids_preferences_frame;
1196     call ShowFrame;
1197 end;
1198
1199 function SaveAndEnterAnother;
1200     variables;
1201     numeric nKey;
1202     numeric nReturnCode;
1203     end;
1204     nKey = GetInterventionKeyId;
1205     call MoveInterventionToDBFields;
1206     nReturnCode = SaveInterventionRecord(nKey, sWirelessURL);
1207     call HideFrame;
1208     call ResetFieldsForNextIntervention;
1209     nCurrentFrame = 1;
1210     call ShowFrame;
1211     show intervention_date_button;
1212 end;
1213
1214 function DeleteChosenIntervention;
1215     variables;
1216     numeric fDeleted;
1217     end;
1218     if (message_box(1,"Delete Intervention", "Are you sure you want to delete the selected
1219         intervention?", "Yes","Cancel","", "") = 0);
1220         # delete the record corresponding to the key .
1221         fDeleted = DeleteInterventionRecord(nDBPIDSKeyField);
1222         #
1223         if (fDeleted);

```

```

1223         nTotalSearchableRecords = nTotalSearchableRecords - 1;
1224         if (nTotalSearchableRecords > 0);
1225             if (nCurrentDisplayedRecord > nTotalSearchableRecords);
1226                 nCurrentDisplayedRecord = nCurrentDisplayedRecord - 1;
1227             end_if;
1228             call UpdateSearchDisplay(nCurrentDisplayedRecord);
1229             call PopulateSearchScreen(nCurrentDisplayedRecord - 1);
1230         else;
1231             call ClearSearchDisplay;
1232         end_if;
1233     end_if;
1234 end_if;
1235 end;
1236
1237 function ContinueEditing;
1238     call HideFrame;
1239     nCurrentFrame = 1;
1240     call ShowFrame;
1241 end;
1242
1243 function CancelPreferences;
1244     hide pids_preferences_frame;
1245     call ShowFrame;
1246 end;
1247
1248 #
1249 #-----
1250 # Functions related to SEARCH text entry screens...
1251 #-----
1252 #
1253 function ShowSearchTextEntryScreen;
1254     variables;
1255         numeric nIndex;
1256     end;
1257     for nIndex = 0, nIndex < nTotalTextEntries;
1258         if objSearchTextEntryButtons[nIndex] = invoker;
1259             search_text_entry_frame.display = sTitles[nIndex];
1260             txSearchTextEntry.display = invoker.object_string;
1261         end_if;
1262         next nIndex + 1;
1263         hide intervention_search_frame;
1264         show search_text_entry_frame;
1265     end;
1266
1267 function OkSearchTextEntry;
1268     hide search_text_entry_frame;
1269     show intervention_search_frame;
1270 end;
1271
1272 function EditIntervention;
1273     fUpdateMode = true;
1274     hide intervention_search_frame;
1275     nCurrentFrame = 1;
1276     call ShowFrame;
1277     call MoveDBFieldsToIntervention;
1278     show intervention_date_button;
1279 end;
1280
1281 function ExitSearch;
1282     hide intervention_search_frame;
1283     call ShowFrame;
1284 end;
1285
1286 #
1287 #-----
1288 #          STARTUP AND SHUTDOWN INVOKER FUNCTIONS
1289 #-----
1290 #
1291 # invoked at program startup
1292 #
1293

```

```
1294 function startup;
1295     call Init;
1296     nCurrentFrame = 0;
1297     call ResetFieldsForNextIntervention;
1298     if (length(sWirelessURL) > 0);
1299         show PIDS_intro_frame;
1300     else;
1301         call miPreferences;
1302     end_if;
1303 end;
1304
1305 #
1306 # invoked at program shutdown,
1307 # closes pids and preferences db
1308 #
1309 function shutdown;
1310     call dba_CloseDB(dbPids);
1311     call dba_CloseDB(dbPidsPrefs);
1312 end;
1313
1314
```

```

1315 #
1316 # CASL_NET_ERRORS.CPK - Package containing CASL / PalmOS error codes/functions (Version
1317 # 3.2)
1318 #
1319 # This file is a CASL package intended for inclusion in CASL programs
1320 # that will make use of the PalmOS networking error codes and their associated
1321 # strings.
1322 #
1323 # NOTE: The information regarding network error code number to string mapping
1324 # was derived from the PalmOS SDK files Errorbase.h, NetMgr.h and InetMgr.h
1325 #
1326 # This file expects to be included in CASL applications after
1327 # the CASLERR2.CPK file.
1328 #
1329 variables;
1330 #
1331 # Error code values base values for NetLib and InetLib
1332 #
1333 # base for NetLib (0x1200, NetLibErrorClass from Errorbase.h)
1334 #
1335 numeric cne_net_error_base = 4608;
1336 #
1337 # base for InetLib (0x1400, InetLibErrorClass from Errorbase.h)
1338 #
1339 numeric cne_inet_error_base = 5120;
1340 #
1341 #
1342 # Strings associated with each NetLib error code
1343 #
1344 numeric cne_number_of_net_error_strings = 82;
1345 #
1346 cne_net_error_strings[cne_number_of_net_error_strings] =
1347 #
1348 "NoError", # (netErrorClass 0)
1349 "AlreadyOpen", # (netErrorClass 1)
1350 "NotOpen", # (netErrorClass 2)
1351 "StillOpen", # (netErrorClass 3)
1352 "ParamErr", # (netErrorClass 4)
1353 "NoMoreSockets", # (netErrorClass 5)
1354 "OutOfResources", # (netErrorClass 6)
1355 "OutOfMemory", # (netErrorClass 7)
1356 "SocketNotOpen", # (netErrorClass 8)
1357 "SocketBusy", # (netErrorClass 9)
1358 "MessageTooBig", # (netErrorClass 10)
1359 "SocketNotConnected ", # (netErrorClass 11)
1360 "NoInterfaces", # (netErrorClass 12)
1361 "BufTooSmall", # (netErrorClass 13)
1362 "Unimplemented", # (netErrorClass 14)
1363 "PortInUse", # (netErrorClass 15)
1364 "QuietTimeNotRelapsed", # (netErrorClass 16)
1365 "Internal", # (netErrorClass 17)
1366 "Timeout", # (netErrorClass 18)
1367 "SocketAlreadyConnected ", # (netErrorClass 19)
1368 "SocketClosedByRemote ", # (netErrorClass 20)
1369 "OutOfCmdBlocks", # (netErrorClass 21)
1370 "WrongSocketType", # (netErrorClass 22)
1371 "SocketNotListening", # (netErrorClass 23)
1372 "UnknownSetting", # (netErrorClass 24)
1373 "InvalidSettingSize", # (netErrorClass 25)
1374 "PrefNctFound", # (netErrorClass 26)
1375 "InvalidInterface", # (netErrorClass 27)
1376 "InterfaceNotFound", # (netErrorClass 28)
1377 "TooManyInterfaces", # (netErrorClass 29)
1378 "BufWrongSize", # (netErrorClass 30)
1379 "UserCancel", # (netErrorClass 31)
1380 "BadScript", # (netErrorClass 32)
1381 "NoSocket ", # (netErrorClass 33)
1382 "SocketRcvBufFull", # (netErrorClass 34)
1383 "NoPendingConnect", # (netErrorClass 35)
1384 "UnexpectedCmd", # (netErrorClass 36)

```

```

1385 "NoTCB", # (netErrorClass 37)
1386 "NilRemoteWindowSize", # (netErrorClass 38)
1387 "NoTimerProc", # (netErrorClass 39)
1388 "SocketInputShutdown", # (netErrorClass 40)
1389 "CmdBlockNotCheckedCut", # (netErrorClass 41)
1390 "CmdNotDone", # (netErrorClass 42)
1391 "UnknownProtocol", # (netErrorClass 43)
1392 "UnkrownService", # (netErrorClass 44)
1393 "UnreachableDest", # (netErrorClass 45)
1394 "ReadOnlySetting", # (netErrorClass 46)
1395 "WouldBlock", # (netErrorClass 47)
1396 "AlreadyInProgress", # (netErrorClass 48)
1397 "PPPTIMEOUT", # (netErrorClass 49)
1398 "PPPBroughtDown", # (netErrorClass 50)
1399 "AuthFailure", # (netErrorClass 51)
1400 "PPPAddressRefused", # (netErrorClass 52)
1401 "NameTooLong", # (netErrorClass 53)
1402 "DNSBadName", # (netErrorClass 54)
1403 "DNSBadArgs", # (netErrorClass 55)
1404 "DNSLabelTooLong", # (netErrorClass 56)
1405 "DNSAllocationFailure", # (netErrorClass 57)
1406 "DNSTimeout", # (netErrorClass 58)
1407 "DNSUnreachable", # (netErrorClass 59)
1408 "DNSFormat", # (netErrorClass 60)
1409 "DNSServerFailure", # (netErrorClass 61)
1410 "DNSNonexistentName", # (netErrorClass 62)
1411 "DNSNIX", # (netErrorClass 63)
1412 "DNSRefused", # (netErrorClass 64)
1413 "DNSImpossible", # (netErrorClass 65)
1414 "DNSNoRRS", # (netErrorClass 66)
1415 "DNSAborted", # (netErrorClass 67)
1416 "DNSBadProtocol", # (netErrorClass 68)
1417 "DNSTruncated", # (netErrorClass 69)
1418 "DNSNoRecursion", # (netErrorClass 70)
1419 "DNSIrrelevant", # (netErrorClass 71)
1420 "DNSNotInLocalCache", # (netErrorClass 72)
1421 "DNSNoPort", # (netErrorClass 73)
1422 "IPCantFragment", # (netErrorClass 74)
1423 "IPNoRoute", # (netErrorClass 75)
1424 "IPNoSrc", # (netErrorClass 76)
1425 "IPNoDst", # (netErrorClass 77)
1426 "IPKtOverflow", # (netErrorClass 78)
1427 "TooManyTCPConnections", # (netErrorClass 79)
1428 "NoDNSServers", # (netErrorClass 80)
1429 "InterfaceDown" # (netErrorClass 81)
1430 ;
1431
1432
1433 #
1434 # Strings associated with each InetLib error code
1435 #
1436 numeric cne_number_of_inet_error_strings = 96;
1437
1438 string cne_inet_error_strings[cne_number_of_inet_error_strings] =
1439
1440 "NoError", # InetLibErrorClass 00
1441 "TooManyClients", # InetLibErrorClass 01
1442 "HandleInvalid", # InetLibErrorClass 02
1443 "ParamsInvalid", # InetLibErrorClass 03
1444 "URLVersionInvalid", # InetLibErrorClass 04
1445 "URLBufTooSmall", # InetLibErrorClass 05
1446 "URLInvalid", # InetLibErrorClass 06
1447 "TooManySockets", # InetLibErrorClass 07
1448 "NoRequestCreated", # InetLibErrorClass 08
1449 "NotConnected", # InetLibErrorClass 09
1450 "InvalidRequest", # InetLibErrorClass 10
1451 "NeedTime", # InetLibErrorClass 11
1452 "HostnameInvalid", # InetLibErrorClass 12
1453 "InvalidPort", # InetLibErrorClass 13
1454 "InvalidHostAddr", # InetLibErrorClass 14
1455 "NilBuffer", # InetLibErrorClass 15

```


| | | | |
|------|-----------------------------------|---------------------|----|
| 1456 | "ConnectTimeout", | # InetLibErrorClass | 16 |
| 1457 | "ResolveTimeout", | # InetLibErrorClass | 17 |
| 1458 | "SendReqTimeout", | # InetLibErrorClass | 18 |
| 1459 | "ReadTimeout", | # InetLibErrorClass | 19 |
| 1460 | "BufTooSmall", | # InetLibErrorClass | 20 |
| 1461 | "SchemeNotSupported", | # InetLibErrorClass | 21 |
| 1462 | "InvalidResponse", | # InetLibErrorClass | 22 |
| 1463 | "Empty", | | |
| 1464 | "Empty", | | |
| 1465 | "SettingTooLarge", | # InetLibErrorClass | 25 |
| 1466 | "SettingSizeInvalid", | # InetLibErrorClass | 26 |
| 1467 | "RequestTooLong", | # InetLibErrorClass | 27 |
| 1468 | "SettingNotImplemented", | # InetLibErrorClass | 28 |
| 1469 | "ConfigNotFound", | # InetLibErrorClass | 29 |
| 1470 | "ConfigCantDelete", | # InetLibErrorClass | 30 |
| 1471 | "ConfigTooMany", | # InetLibErrorClass | 31 |
| 1472 | "ConfigBadName", | # InetLibErrorClass | 32 |
| 1473 | "ConfigNotAlias", | # InetLibErrorClass | 33 |
| 1474 | "ConfigCantPointToAlias", | # InetLibErrorClass | 34 |
| 1475 | "ConfigEmpty", | # InetLibErrorClass | 35 |
| 1476 | "Empty", | | |
| 1477 | "ConfigAliasErr", | # InetLibErrorClass | 37 |
| 1478 | "NoWirelessInterface", | # InetLibErrorClass | 38 |
| 1479 | "EncryptionNotAvail", | # InetLibErrorClass | 39 |
| 1480 | "NeedRetryEncSegNum", | # InetLibErrorClass | 40 |
| 1481 | "NeedRetryEncPublicKey", | # InetLibErrorClass | 41 |
| 1482 | "ResponseTooShort", | # InetLibErrorClass | 42 |
| 1483 | "MobitexIllegalOKHost", | # InetLibErrorClass | 43 |
| 1484 | "MobitexIllegalBadHost", | # InetLibErrorClass | 44 |
| 1485 | "HTTPBadRequest", | # InetLibErrorClass | 45 |
| 1486 | "HTTPUnauthorized", | # InetLibErrorClass | 46 |
| 1487 | "HTTPForbidden", | # InetLibErrorClass | 47 |
| 1488 | "HTTPNotFound", | # InetLibErrorClass | 48 |
| 1489 | "HTTPMethodNotAllowed", | # InetLibErrorClass | 49 |
| 1490 | "HTTPNotAcceptable", | # InetLibErrorClass | 50 |
| 1491 | "HTTPProxyAuthRequired", | # InetLibErrorClass | 51 |
| 1492 | "HTTPRequestTimeout", | # InetLibErrorClass | 52 |
| 1493 | "HTTPConflict", | # InetLibErrorClass | 53 |
| 1494 | "HTTPGone", | # InetLibErrorClass | 54 |
| 1495 | "HTTPLengthRequired", | # InetLibErrorClass | 55 |
| 1496 | "HTTPPreconditionFailed", | # InetLibErrorClass | 56 |
| 1497 | "HTTPRequestTooLarge", | # InetLibErrorClass | 57 |
| 1498 | "HTTPRequestURITooLong", | # InetLibErrorClass | 58 |
| 1499 | "HTTPUnsupportedType", | # InetLibErrorClass | 59 |
| 1500 | "HTTPServerError", | # InetLibErrorClass | 60 |
| 1501 | "CTPServerError", | # InetLibErrorClass | 61 |
| 1502 | "TypeNotCached", | # InetLibErrorClass | 62 |
| 1503 | "ErrCacheInvalid", | # InetLibErrorClass | 63 |
| 1504 | "URLDispatched", | # InetLibErrorClass | 64 |
| 1505 | "DatabaseNotFound", | # InetLibErrorClass | 65 |
| 1506 | "CTPMalformedRequest", | # InetLibErrorClass | 66 |
| 1507 | "CTPUnknownCommand", | # InetLibErrorClass | 67 |
| 1508 | "CTPTruncated", | # InetLibErrorClass | 68 |
| 1509 | "CTPUnknownError", | # InetLibErrorClass | 69 |
| 1510 | "CTPProxyError", | # InetLibErrorClass | 70 |
| 1511 | "CTPSocketErr", | # InetLibErrorClass | 71 |
| 1512 | "CTPInvalidURL", | # InetLibErrorClass | 72 |
| 1513 | "CTPReferringPageOutOfDate", | # InetLibErrorClass | 73 |
| 1514 | "CTPBadRequest", | # InetLibErrorClass | 74 |
| 1515 | "UNUSED", | # InetLibErrorClass | 75 |
| 1516 | "CTPMailServerDown", | # InetLibErrorClass | 76 |
| 1517 | "CTPHostNotFound", | # InetLibErrorClass | 77 |
| 1518 | "CTPContentInvalidTag", | # InetLibErrorClass | 78 |
| 1519 | "CTPContentInternal", | # InetLibErrorClass | 79 |
| 1520 | "CTPContentDataEnd", | # InetLibErrorClass | 80 |
| 1521 | "CTPContentResourceTooBig", | # InetLibErrorClass | 81 |
| 1522 | "CTPContentNoFrames", | # InetLibErrorClass | 82 |
| 1523 | "CTPContentUnsupportedContent", | # InetLibErrorClass | 83 |
| 1524 | "CTPContentUnsupportedEncoding", | # InetLibErrorClass | 84 |
| 1525 | "CTPContentBadForm", | # InetLibErrorClass | 85 |
| 1526 | "CTPContentBadFormMissingAction", | # InetLibErrorClass | 86 |

```

1527 "CTPContentBagFormMissingMethod", # InetLibErrorClass 87
1528 "CTPContentNoSourceData", # InetLibErrorClass 88
1529 "CTPContentBadImage", # InetLibErrorClass 89
1530 "CTPContentImageTooLarge", # InetLibErrorClass 90
1531 "MobitexErrorHandled", # InetLibErrorClass 91
1532 "ProxyDownBadHost", # InetLibErrorClass 92
1533 "HostConnectionLost", # InetLibErrorClass 93
1534 "LinkNotFound", # InetLibErrorClass 94
1535 ;
1536
1537 #
1538 # these networking error values are generated by the CASLrt or
1539 # the CASLpro support library
1540 #
1541 numeric cne_socket_create = -51;
1542 numeric cne_protocol_spec = -52;
1543 numeric cne_missing_node = -53;
1544 numeric cne_protocol_not_found = -54;
1545 numeric cne_host_lookup = -55;
1546 numeric cne_no_connection = -56;
1547 numeric cne_socket_closed = -57;
1548
1549 cne_string = ""; # CASL networking error string
1550
1551 end;
1552
1553 #
1554 # function to put appropriate error string into the variable "cne_string"
1555 # for use by programs including this CASL package.
1556 #
1557 function cne_geterror as string;
1558 if errorcode > 0;
1559
1560 compile_if platform = "pilot";
1561
1562 if (errorcode >= cne_net_error_base)
1563 and (errorcode < (cne_net_error_base + cne_number_of_net_error_strings) );
1564 cne_geterror = cne_net_error_strings[errorcode - cne_net_error_base];
1565 else_if (errorcode >= cne_inet_error_base)
1566 and (errorcode < (cne_inet_error_base + cne_number_of_inet_error_strings) );
1567 cne_geterror = cne_inet_error_strings[errorcode - cne_inet_error_base];
1568 else;
1569 cne_geterror = "OS error: " + string(errorcode, "");
1570 end_if;
1571
1572 compile_else;
1573
1574 cne_geterror = "OS error: " + string(errorcode, "");
1575
1576 compile_end_if;
1577
1578 else_if errorcode = 0;
1579 cne_geterror = "NoError";
1580 else_if errorcode = cne_socket_create;
1581 cne_geterror = "SocketCreatError";
1582 else_if errorcode = cne_protocol_spec;
1583 cne_geterror = "BadProtocol";
1584 else_if errorcode = cne_missing_node;
1585 cne_geterror = "MissingHost";
1586 else_if errorcode = cne_protocol_not_found;
1587 cne_geterror = "ProtocolNotFound";
1588 else_if errorcode = cne_host_lookup;
1589 cne_geterror = "HostNotFound";
1590 else_if errorcode = cne_no_connection;
1591 cne_geterror = "NoConnection";
1592 else_if errorcode = cne_socket_closed;
1593 cne_geterror = "SocketClosed";
1594 else;
1595 cne_geterror = ce_errors[-errorcode];
1596 end_if;
1597 end;

```

1598

1599

```

1600 #
1601 # CASL_Palm_Utils.cpk
1602 #
1603 # CASL interface definition to C functions to perform PalmOS operations
1604 # not currently directly supported in CASL.
1605 #
1606 # This file defines the call interfaces to callable C functions that reside
1607 # in the CASL_Palm_Utils library. The function definitions are wrapped inside
1608 # the EXTERNAL specification of CASL, indicating that code for the functions
1609 # themselves is resident in the external library.
1610 #
1611 # Copyright(c) 2001 Feras Information Technologies
1612 # Use of this code is permitted for any purpose, provided
1613 # you leave the FIT copyright notice, and you accept code
1614 # "AS IS" without warranty of any kind.
1615 #
1616 #
1617 # ver 2.6.0.2, 7jan01, frank o'brien, caslsoft
1618 # support@caslsoft.com, www.caslsoft.com
1619 # added supplementary usage comments and CASL_FixTimeBug function
1620 #
1621 # ver 2.6.0.1, 22oct98, joan feras, caslsoft
1622 # original file
1623 #
1624 external "CASL_Palm_Utils";
1625 #
1626 # CASL_SelectDay
1627 # Inputs are clear.
1628 # When presented with dialog, user can tap OK or Cancel.
1629 # If user taps OK, the return string is date in format "YYYYMMDD".
1630 # If user taps Cancel, the return string is "" (null string).
1631 #
1632 function CASL_SelectDay # display and manipulate the day pick dialog
1633 (
1634     string title, # title for the date picking form
1635     string day_string # string containing start date using format YYYYMMDD
1636 ) as string; # returns chosen date using same format
1637 #
1638 #
1639 # CASL_SelectTime
1640 # The 'times' input array must be preloaded with values that make sense.
1641 # Time format is HHMM, 24 h format string, examples: 6:15a="0615",
1642 # 6:05p="1805", midnight="0000", noon="1200".
1643 # The start time element of array serves as default highlighted
1644 # value of time selector. Array must have at least 2 elements, even
1645 # if you want only 1 time, you could get some crashes if you pass
1646 # a 2nd null string or no 2nd element. If you pass 2 null strings,
1647 # the dialog won't show and the return value will be 0 (Cancel).
1648 # The 'start_of_day' input is an integer numeric of what the time
1649 # dialog selector should start with. for example passing 6,
1650 # will have 6:00a as first visible time without needing scroll arrows,
1651 # 18 will have 6:00p. It's smart enough to override this setting if
1652 # needed to show the inputted start 'times' value.
1653 # When presented with dialog, user can tap OK or Cancel.
1654 # If user taps OK, the return numeric is 1. This does not necessarily
1655 # mean that the time changed, it could still be default,
1656 # it only means OK has tapped. Returned 'times' elements are the new
1657 # selected times.
1658 # If user taps Cancel, the return numeric is 0. It appears that the
1659 # passed 'times' elements are returned unchanged even if user had
1660 # different times selected before tapping cancel.
1661 # With CASL_SelectTime, when return time is 00mm time, the actual
1662 # return value will be 0mm, when return time is 000m, the actual
1663 # return value will be 0m, this is bug in Palm API pick time dialog,
1664 # for return times between 12 midnight (0000) and 12:09a (0009),
1665 # to get the correct 4 digit return time, hhmm, including any
1666 # preceeding 0's, use CASL_FixTimeBug
1667 #
1668 function CASL_SelectTime # display and manipulate the time pick dialog
1669 (
1670     string times[], # string array with start/end times at cells 0 and 1

```

```
1671 string title,           # title for the time picking form
1672 numeric start_of_day     # top hour displayed in hour list
1673 ) as numeric;           # returns 1 if user taps ok, 0 otherwise
1674
1675 # CASL_SysGetRomToken
1676 ### John, you may want to add some comments here,
1677 ### not sure if doesn't work on Palm VII??
1678
1679 # get the serial number of the PalmIII ROM (if possible)
1680 #
1681 function CASL_SysGetRomToken as string;
1682
1683 end_external;
1684
1685 # CASL_FixTimeBug
1686 # given time in format hhmm, or 0mm, or 0m,
1687 # such as an element of the 'times' array from CASL_SelectTime,
1688 # returns hhmm,
1689 # hhmm -> hhmm
1690 # 0mm -> 00mm
1691 # 0m -> 000m
1692
1693 function CASL_FixTimeBug(string time) as string;
1694     if length(time)=4;
1695         CASL_FixTimeBug=time;
1696     else;
1697         if length(time)=3;
1698             CASL_FixTimeBug="0"+time;
1699         else;
1700             CASL_FixTimeBug="0"+left(time,1)+"0"+right(time,1);
1701         end_if;
1702     end_if;
1703 end;
1704
1705
```

```
1706 #
1707 # CASLER26.CPK - Package containing CASL 2.0 error codes/functions
1708 #
1709 # This file is a CASL package intended for inclusion in CASL programs
1710 # that will make use of the interpreter error codes and their associated
1711 # strings.
1712 #
1713 # NOTE: this file is designed specifically for use with CASL version 2.0
1714 # due to its usage function return values, and the "bad number"
1715 # error code.
1716 #
1717 # Copyright(c) 2001 Feras Information Technologies
1718 # Use of this code is permitted for any purpose, provided
1719 # you leave the FIT copyright notice, and you accept code
1720 # "AS IS" without warranty of any kind.
1721 #
1722 #
1723 # Modifications by Scott Smith, March 1999 to include ce_no_error and to
1724 # remove reference to ce_string variable that is no longer used/needed.
1725 #
1726 variables;
1727 #
1728 # Error code values
1729 #
1730 #
1731 ce_no_error = 0;
1732 ce_div_0 = -1;
1733 ce_array = -2;
1734 ce_not_open = -3;
1735 ce_fld_mis = -4;
1736 ce_eof = -5;
1737 ce_fmode = -6;
1738 ce_file_nf = -7;
1739 ce_file_del = -8;
1740 ce_serial_open = -9;
1741 ce_serial_comm = -10;
1742 ce_file_open = -11;
1743 ce_rec_not_found = -12;
1744 ce_put_record_error = -13;
1745 ce_file_delete_error = -14;
1746 ce_bad_number_error = -15;
1747 #
1748 # Strings associated with each error code
1749 #
1750 #
1751 ce_errors[16] = "No Error", "Divide by 0",
1752 "Invalid array index", "File not open",
1753 "DB field mismatch", "File EOF",
1754 "Invalid file mode", "File not found",
1755 "Record deleted", "Serial port error",
1756 "Serial comm error", "File already open",
1757 "Record not found", "Put record error",
1758 "Delete record error", "Bad Number Error";
1759 #
1760 #
1761 # CASL error string (no longer used)
1762 #
1763 #
1764 # ce_string = "";
1765 #
1766 end;
1767 #
1768 #
1769 # function to put appropriate error string into the variable "ce_string"
1770 # for use by programs including this CASL package.
1771 #
1772 #
1773 function ce_geterror as string;
1774 #
1775 variables;
```

```
1777     numeric temp;
1778     end;
1779
1780     if errorcode > 0;
1781         ce_geterror = "OS error: " + string(errorcode, "");
1782     else;
1783         ce_geterror = ce_errors[-errorcode];
1784     end_if;
1785
1786     if false();
1787         # just to prevent "var not used" warning messages
1788         temp = ce_no_error + ce_div_0 + ce_array + ce_not_open + ce_fld_mis
1789             + ce_eof + ce_fmode + ce_file_nf + ce_file_del + ce_serial_open
1790             + ce_serial_comm + ce_file_open + ce_rec_not_found
1791             + ce_put_record_error + ce_file_delete_error + ce_bad_number_error;
1792     end_if;
1793
1794     end;
1795
1796
```

```

1797 #
1798 # dba_DBAccess.cpk,
1799 # generic dbfile functions
1800 #
1801 # Copyright(c) 2001 Feras Information Technologies
1802 # Use of this code is permitted for any purpose, provided
1803 # you leave the FIT copyright notice, and you accept code
1804 # "AS IS" without warranty of any kind.
1805 #
1806 #
1807 # ver 3.1.0.1, 7jan01, frank o'brien, caslsoft
1808 # fobrien@caslsoft.com, www.caslsoft.com
1809 #
1810 # this file contains all basic functions needed to work with
1811 # with dbfiles.
1812 # functions are provided to: create/open db, existing
1813 # record check and select, write record, read exact record,
1814 # read bestfit record, remove record.
1815 # all these functions return true/false values for
1816 # success. this makes it easy to do errorchecking
1817 # and leaves error messages to the programmer,
1818 # these functions are defined without a return value:
1819 # close db, delete db.
1820 #
1821 # depending on dbfile usage, it's sometimes useful to fill an array with
1822 # all key field strings in the dbfile to ease record selection,
1823 # no generic function is provided in this package, as the bound
1824 # field variables are specific to each dbfile. this task is left
1825 # to programmer. normally this is done similar to this:
1826 #
1827 # i=0;
1828 # errorcode=0;
1829 # get_fields dbData;
1830 # while errorcode=0;
1831 #   sKeyList[i]=sKey;
1832 #   i=i+1;
1833 #   get_fields dbData;
1834 # end_while;
1835 # if errorcode<>-5; # eof
1836 #   # show error
1837 # else;
1838 #   i is 1-based record count, all keys are in sKeyList array
1839 # end_if;
1840 #
1841 # for dbfile that is used on both palm and windows platforms,
1842 # with merge sync_pref, and when same record is changed on
1843 # each platform between hotsync, can get duplicate records, and/or
1844 # number of records exceeding max allowed (if there is a limit),
1845 # sometimes may want these checks in fill loop too.
1846 #
1847 # create/open db
1848 #
1849 # given dbfile object, and filename,
1850 # sideeffects: if db does not exist, db will
1851 # be created with filename and record structure
1852 # defined by bound field variables, if db
1853 # exists, opens db with filename, errorcode has error status,
1854 # if desired, use CASL file function "Exists" to check if
1855 # db file exists prior to call,
1856 # returns true/false with success
1857 #
1858 function dba_ffOpenDB(object dbobj,string dbname) as numeric;
1859   errorcode=0;
1860   open dbobj,dbname;
1861   dba_ffOpenDB=errorcode=0;
1862 end;
1863 #
1864 # given dbfile object, and key field data string,
1865 # side effect: db pointer set to matching record if found,
1866 # or record right after in sort order if not found,
1867 # errorcode has error status

```



```
1868 # returns true if key is unique
1869 # returns false if key was found, or unexpected error
1870
1871 function dba_ffRecordExists(object dbobj,numeric key) as numeric;
1872     errorcode=0;
1873     search dbobj,key;
1874     dba_ffRecordExists=errorcode=0;
1875 end;
1876
1877 # write record
1878
1879 # db bound variables already have field data
1880 # given dbfile object, and key field data string,
1881 # side effect: if key is new insert new record, if key
1882 # already exists writes new data for existing record,
1883 # errorcode has error status
1884 # returns true/false on success
1885
1886 function dba_ffWriteRecord(object dbobj,numeric key) as numeric;
1887     if dba_ffRecordExists(dbobj,key);
1888         # write existing record
1889         errorcode=0;
1890         put_fields dbobj;
1891     else;
1892         # add new record
1893         errorcode=0;
1894         insert dbobj;
1895     end_if;
1896     dba_ffWriteRecord=errorcode=0;
1897 end;
1898
1899 # read record, 2 versions
1900
1901 # given dbfile object, and key field data string,
1902 # side effect: fills db bound variables with field data
1903 # of record exactly matching key field data string,
1904 # errorcode has error status
1905 # returns true/false on success
1906
1907 function dba_ffReadExactRecord(object dbobj,numeric key) as numeric;
1908     if dba_ffRecordExists(dbobj,key);
1909         # found
1910         errorcode=0;
1911         get_fields dbobj;
1912         dba_ffReadExactRecord=errorcode=0;
1913     else;
1914         # not found
1915         dba_ffReadExactRecord=false;
1916     end_if;
1917 end;
1918
1919
1920 # given dbfile object, and key field data string,
1921 # side effect: fills db bound variables with field
1922 # data from matching record if found,
1923 # or record right after in sort order if not found,
1924 # errorcode has error status
1925 # returns true/false on success
1926
1927 function dba_ffReadBestFitRecord(object dbobj,numeric key) as numeric;
1928     if dba_ffRecordExists(dbobj,key);
1929         # found
1930         errorcode=0;
1931         get_fields dbobj;
1932         dba_ffReadBestFitRecord=errorcode=0;
1933     else;
1934         # not found, read closest fit
1935         errorcode=0;
1936         get_fields dbobj;
1937         dba_ffReadBestFitRecord=errorcode=0;
1938     end_if;
```

```
1939 end;
1940
1941
1942 # given dbfile object, and a record number,
1943 # side effect: fills db bound variables with field data
1944 # errorcode has error status
1945 # returns true/false on success
1946 function dba_ffReadRecordFromRecNo(object dbobj, numeric recNo) as numeric;
1947     errorcode=0;
1948     seek_start dbobj, recNo;
1949
1950     if errorcode=0;
1951         # found
1952         errorcode=0;
1953         get_fields dbobj;
1954         dba_ffReadRecordFromRecNo=errorcode=0;
1955     else;
1956         # not found
1957         dba_ffReadRecordFromRecNo=false;
1958     end_if;
1959 end;
1960
1961
1962 # remove record
1963
1964 # given dbfile object, and key field data string,
1965 # side effect: removes record with matching key field
1966 # data string from db, errorcode has error status,
1967 # returns true/false on success, returns true if doesn't exist
1968
1969 function dba_ffRemoveRecord(object dbobj,numeric key) as numeric;
1970     if dba_ffRecordExists(dbobj,key);
1971         errorcode=0;
1972         remove dbobj;
1973         dba_ffRemoveRecord=errorcode=0;
1974     else;
1975         dba_ffRemoveRecord=true;
1976     end_if;
1977 end;
1978
1979 # close db
1980
1981 # given dbfile object,
1982 # sideeffect: closes dbfile, errorcode has error status,
1983 # it's assumed programmer will normally not be able to do
1984 # anything about failure, so there's no return value,
1985 # if programmer is interested in error status, the
1986 # system's errorcode value can be queried after call,
1987 # in this way, programmer could add a "retry/abort" option.
1988
1989 function dba_CloseDB(object dbobj);
1990     errorcode=0;
1991     close dbobj;
1992 end;
1993
1994 # delete db
1995
1996 # given dbfile object, and filename,
1997 # sideeffect: deletes dbfile, dbfile will be closed
1998 # before delete if it's open, errorcode has error status,
1999 # it's assumed programmer will normally not be able to do
2000 # anything about failure, so there's no return value,
2001 # if programmer is interested in error status, the
2002 # system's errorcode value can be queried after call,
2003 # in this way, programmer could add a "retry/abort" option.
2004
2005 function dba_DeleteDB(object dbobj,string dbname);
2006     errorcode=0;
2007     if Exists(dbname);
2008         dba_CloseDB(dbobj); # can't delete open file
2009         delete dbname;
```

```
2010     end_if;  
2011 end;  
2012  
2013
```

```

2014 #
2015 # fdt_FormatDateTime.cpk,
2016 # gregorian date, weekday, and time formatting functions
2017 #
2018 # Copyright(c) 2001 Feras Information Technologies
2019 # Use of this code is permitted for any purpose, provided
2020 # you leave the FIT copyright notice, and you accept code
2021 # "AS IS" without warranty of any kind.
2022 #
2023
2024 # see also:
2025 #   jdt_JulianDateTime.cpk for julian date and time functions,,
2026 #   gdt_GregorianDateTime.cpk for gregorian date and time functions
2027
2028 # ver 3.1.0.2, 21jan01, frank o'brien, caslsoft
2029 # support@caslsoft.com, www.caslsoft.com
2030
2031 # given Gregorian date in format, YYYYMMDD,
2032 # where YYYY=-4900 to 9999, MM=month=01-12, DD=day=01-31,
2033 # and dateformat string in form "dd mmm yy", or "dd mmm yyyy",
2034 # or "mm/dd/yy" or "dd.mm.yy",
2035 # dateformat string is case-insensitive,
2036 # returns Gregorian date in format specified, where
2037 # dd=days without leading zero
2038 # mm=month without leading zero
2039 # mmm=month of form Jan-Dec,
2040 # yy=last 2 digits of year with leading zero
2041 # yyyy=year
2042 # if dateformat string is unrecognized, date is returned in format
2043 # "dd mmm yy",
2044 # no error checking
2045
2046 function fdt_sfFormatDate(string gd,string dateformat) as string;
2047     variables;
2048         numeric y;
2049         numeric m;
2050         numeric d;
2051     end;
2052     y=value(left(gd,length(gd)-4));
2053     m=value(mid(gd,length(gd)-4,2));
2054     d=value(right(gd,2));
2055     if(lower(dateformat)="mm/dd/yyyy");
2056         # format of form 3/17/2001
2057         fdt_sfFormatDate=
2058             string(m,"")+ "/" +
2059             string(d,"")+ "/" +
2060             string(y,"");
2061     else;
2062         if lower(dateformat)="mm/dd/yy";
2063             # format of form 3/17/01
2064             fdt_sfFormatDate=
2065                 string(m,"")+ "/" +
2066                 string(d,"")+ "/" +
2067                 string(y%100,"0#");
2068         else;
2069             if lower(dateformat)="dd.mm.yy";
2070                 # format of form 17.3.01
2071                 fdt_sfFormatDate=
2072                     string(d,"")+ ". " +
2073                     string(m,"")+ ". " +
2074                     string(y%100,"C#");
2075             else;
2076                 if lower(dateformat)="dd mmm yyyy";
2077                     # format of form 17 Mar 2001
2078                     fdt_sfFormatDate=
2079                         string(d,"")+ " " +
2080                         mid("JanFebMarAprMayJunJulAugSepOctNovDec", (m-1)*3,3)+ " " +
2081                         string(y,"");
2082                 else;
2083                     # format of form 17 Mar 01, or unrecognized format
2084                     fdt_sfFormatDate=

```

```

2085         string(d,"")+ " "+
2086         mid("JanFebMarAprMayJunJulAugSepOctNovDec", (m-1)*3,3)+ " "+
2087         string(y%100,"0#");
2088     end_if;
2089 end_if;
2090 end_if;
2091 end_if;
2092 end;
2093
2094 # given Gregorian time in format, HHmm, 24 h format.
2095 # and timeformat string in form "hh:mm 24", "hh:mm 12", or "hh.mm",
2096 # timeformat string is case-insensitive,
2097 # returns time in specified format where,
2098 # hh=hours with leading zero, see also 24 and 12
2099 # mm=minutes with leading zero
2100 # 24=hours are in modulo 24
2101 # 12=hours are in modulo 12, suffix is "am" if hours=0-11 in modulo 24,
2102 # and "pm" if hours=12-23 in modulo 24
2103 # if timeformat string is unrecognized, time is returned in format
2104 # "hh:mm 24",
2105 # no error checking
2106
2107 function fdt_sfFormatTime(string gt,string timeformat) as string;
2108     variables;
2109         numeric h;
2110         numeric mn;
2111     end;
2112     h=value(left(gt,2));
2113     mn=value(mid(gt,2,2));
2114     if lower(timeformat)="hh.mm";
2115         # 24 h format, decimal separator, good for led font
2116         fdt_sfFormatTime=string(h,"0#")+ "." +string(mn,"0#");
2117     else;
2118         if lower(timeformat)="hh:mm 12";
2119             # am/pm format
2120             if h/12<1;
2121                 # am
2122                 if h=0;
2123                     fdt_sfFormatTime="12:"+string(mn,"0#")+ " am";
2124                 else;
2125                     fdt_sfFormatTime=string(h,"0#")+ ":" +
2126                     string(mn,"0#")+ " am";
2127                 end_if;
2128             else;
2129                 # pm
2130                 if h=12;
2131                     fdt_sfFormatTime="12:"+string(mn,"0#")+ " pm";
2132                 else;
2133                     fdt_sfFormatTime=string(h%12,"C#")+ ":" +
2134                     string(mn,"0#")+ " pm";
2135                 end_if;
2136             end_if;
2137         else;
2138             # 24 hour format with colon separator, or unrecognized format
2139             fdt_sfFormatTime=string(h,"0#")+ ":" +string(mn,"0#");
2140         end_if;
2141     end_if;
2142 end;
2143
2144 # commented out because OClock sample app would exceed demo limit,
2145 # change false to true if you have registered version, or not
2146 # using with OClock sample app
2147
2148 compile_if false;
2149
2150 # given weekday, 0=mon, ... 6=sun, and wdformat string in form
2151 # of "short" or "long", wdformat is case-insensitive,
2152 # returns string of weekday where,
2153 # if "long", returns "Monday", ... "Sunday"
2154 # if "short" or unrecognized, returns "Mon", ... "Sun"
2155 # no error checking (if wd out of range, should return

```

```
2156 # null string, but not fully tested)
2157
2158 function fdt_sfFormatWeekday(numeric wd,string wdformat) as string;
2159     if lower(wdformat)="long";
2160         # long format
2161         variables;
2162         string days[7]="Monday","Tuesday","Wednesday","Thursday",
2163             "Friday","Saturday","Sunday";
2164     end;
2165     if wd<0 or wd>6;
2166         fdt_sfFormatWeekday="";
2167     else;
2168         fdt_sfFormatWeekday=days[wd];
2169     end_if;
2170 else;
2171     # short format or unrecognized format
2172     fdt_sfFormatWeekday=mid("MonTueWedThuFriSatSun",wd*3,3);
2173 end_if;
2174 end;
2175
2176 compile_end_if;
2177
2178
```

```

2179 #
2180 # jdt_JulianDateTime.cpk,
2181 # julian date, weekday, and time functions,
2182 #
2183 # Copyright(c) 2001 Feras Information Technologies
2184 # Use of this code is permitted for any purpose, provided
2185 # you leave the FIT copyright notice, and you accept code
2186 # "AS IS" without warranty of any kind.
2187 #
2188 #
2189 # see also:
2190 # gdt_GregorianDateTime.cpk for gregorian date and time functions
2191 # fdt_FormatDateTime.cpk for gregorian formatting functions
2192 #
2193 # Algorithm for converting between Gregorian dates and
2194 # Julian Dates presented in 1968 in a letter to the editor of
2195 # Communications of the ACM (CACM, volume 11, number 10,
2196 # October 1968, p.657) by Henry F. Fliegel and
2197 # Thomas C. Van Flandern
2198 #
2199 # Additional information from Peter Meyer at http://hermetic.magnet.ch/
2200 #
2201 # Astronomers and chronologists use a system of numbering days
2202 # called Julian days. The temporal sequence of days is mapped
2203 # onto the sequence of integers, -2, -1, 0, 1, 2, 3, etc. This
2204 # makes it easy to determine the number of days between two days
2205 # (just subtract one Julian day number from the other).
2206 #
2207 # Do not confuse Julian days with the Julian Calender. Julian days
2208 # are a serial date system so all Gregorian dates have a sequential
2209 # ordering to facilitate math involving Gregorian dates
2210 #
2211 # ver 3.1.0.1, 7jan01, frank o'brien, caslsoft
2212 # support@caslsoft.com, www.caslsoft.com
2213 #
2214 # given Gregorian date in format, YYYYMMDD,
2215 # where YYYY=-4900 to 9999, MM=month=01 to 12, DD=day=01 to 31,
2216 # returns the Julian Days, a serial date,
2217 # no error checking,
2218 # handles out of range value as a rollover type thing
2219 # the julian date defaults to midnight (=0)
2220 # no error checking
2221 #
2222 function jdt_nfJulianDate(string gd) as numeric;
2223     variables;
2224         numeric y;
2225         numeric m;
2226         numeric d;
2227     end;
2228     y=value(left(gd,length(gd)-4));
2229     m=value(mid(gd,length(gd)-4,2));
2230     d=value(right(gd,2));
2231     jdt_nfJulianDate=int((1461*(y+4800+int((m-14)/12)))/4)+
2232     int((367*(m-2-12*int((m-14)/12)))/12)-
2233     int((3*int((y+4900+int((m-14)/12))/100))/4)+
2234     d-32075;
2235 end;
2236 #
2237 # given Gregorian time in format, HHmm, 24 h format,
2238 # where HH=hours=00 to 23, and mm=minutes=00 to 59,
2239 # returns decimal equivalent of number of days
2240 # midnight=0.00, noon=0.5, 6:00a=6/24=0.25,
2241 # 6:45p=18/24+45/60/24=a little more than 0.75
2242 # no error checking
2243 #
2244 function jdt_nfJulianTime(string gt) as numeric;
2245     variables;
2246         numeric h;
2247         numeric mn;
2248     end;
2249     h=value(left(gt,2));

```

```
2250     mn=value(mid(gt,2,2));
2251     jdt_nfJulianTime=(h+(mn/60))/24;
2252 end;
2253
2254 # commented out because DaysBetween and OClock sample apps
2255 # would exceed demo limit, change false to true if you have
2256 # registered version, or not using with DaysBetween and
2257 # OClock sample app
2258
2259 compile_if false;
2260
2261 # given Gregorian date in format, YYYYMMDD,
2262 # where YYYY=year=-4900 to 9999, MM=month=01 to 12, DD=day=01 to 31,
2263 # returns modulo 7 of julian date of gregorian date where,
2264 # 0=mon, ... 6=sun
2265 # no error checking
2266
2267 function jdt_nfWeekday(string gd) as numeric;
2268     jdt_nfWeekday=jdt_nfJulianDate(gd)%7;
2269 end;
2270
2271 # given Gregorian date in format, YYYYMMDD,
2272 # where YYYY=year=-4900 to 9999, MM=month=01 to 12, DD=day=01 to 31,
2273 # returns true if inputted gregorian date fits within
2274 # limits given above, returns false otherwise
2275
2276 function jdt_ffLegalDate(string gd) as numeric;
2277     variables;
2278         string y;
2279         string m;
2280         string d;
2281     end;
2282     y=left(gd,length(gd)-4);
2283     m=mid(gd,length(gd)-4,2);
2284     d=right(gd,2);
2285     jdt_ffLegalDate=(y="0000" or (value(y)>=-4900 and value(y)<=9999 and
2286     value(y)<>0)) and (value(m)>=1 and value(m)<=12) and
2287     (value(d)>=1 and value(d)<=31);
2288 end;
2289
2290 # given Gregorian time in format, HHmm, 24 h format,
2291 # where HH=hours=00 to 23, and mn=minutes=00 to 59,
2292 # returns true if inputted gregorian time fits within
2293 # limits given above, returns false otherwise
2294
2295 function jdt_ffLegalTime(string gt) as numeric;
2296     variables;
2297         string h;
2298         string mn;
2299     end;
2300     h=left(gt,2);
2301     mn=mid(gt,2,2);
2302     jdt_ffLegalTime=(h="00" or (value(h)>0 and value(h)<24)) and
2303     (mn="00" or (value(mn)>0 and value(mn)<60));
2304 end;
2305
2306 compile_end_if;
2307
2308
```



```
2309 #
2310 # PIDS_date_lib.cpk
2311 #
2312 #####
2313 #
2314 # PIDS (tm)
2315 # Pharmacy Intervention Documentation System
2316 # Copyright © 2002 Integrated Software Systems, L.L.C.
2317 #
2318 #####
2319 #
2320 # Version: 1.0
2321 # Date: 1/7/2002
2322 # Author: Todd Viegut & Brad Tice
2323 #
2324 #
2325 # Function to handle retrieving the intervention date.
2326 #
2327 include "fdt_FormatDateTime.cpk";
2328 include "jdt_JulianDateTime.cpk";
2329 include "CASL_Palm_Utills.cpk";
2330 #
2331 function GetCurrentInterventionDate as string;
2332     variables;
2333     string sSelectedDate;
2334 end;
2335 sSelectedDate = string(year(),"0###") + string(month(),"0#") + string(day(),"0#");
2336 GetCurrentInterventionDate = fdt_sfFormatDate(sSelectedDate,"mm/dd/yy");
2337 end;
2338
2339 function GetInterventionDate(string sDefaultDate) as string;
2340     compile_if platform = "windows";
2341     variables;
2342     numeric mb;
2343     end;
2344     mb = message_box(2,"Warning","Not intended to run on windows", "", "", "OK");
2345     GetInterventionDate = "";
2346     compile_else;
2347     variables;
2348     string sFormat;
2349     string sSelectedDate;
2350     end;
2351     #
2352     sFormat = string(year(),"0###") + string(month(),"0#") + string(day(),"0#");
2353     #
2354     # running on palm, use external library for date and time
2355     # get new date with date pick dialog
2356     #
2357     sSelectedDate = CASL_SelectDay("Intervention Date", sFormat);
2358     #sSelectedDate = "20020101";
2359     if (sSelectedDate <> "");
2360         GetInterventionDate = fdt_sfFormatDate(sSelectedDate, "mm/dd/yy");
2361     else;
2362         GetInterventionDate = sDefaultDate;
2363     end_if;
2364     compile_end_if;
2365 end;
2366
2367
```

```
2368 #
2369 # PIDS_db_files.cpk
2370 #
2371 #####
2372 #
2373 # PIDS (tm)
2374 # Pharmacy Intervention Documentation System
2375 # Copyright © 2002 Integrated Software Systems, L.L.C.
2376 #
2377 #####
2378 #
2379 # Version: 1.0
2380 # Date: 12/22/2001
2381 # Author: Todd Viegut & Brad Tice
2382 #
2383 #
2384 # Database related variable definitions:
2385 #
2386 variables;
2387
2388 #
2389 #-----
2390 # db fields for preferences...
2391 #-----
2392 #
2393 numeric nDBPrefsKeyField;
2394 numeric fDBPrefsEnableSpeedEntryField;
2395 string sDBPrefsWirelessURLField;
2396 string sDBPrefsLocationIDField;
2397 string sDBPrefsRotationIDField;
2398 string sDBPrefsUserIDField;
2399
2400 #
2401 #-----
2402 # db fields for PIDS...
2403 #-----
2404 #
2405 numeric nDBPIDSKeyField;
2406 numeric nDBSynced;
2407 #
2408 string sDBRotationID;
2409 string sDBLocationID;
2410 string sDBInterventionDate;
2411 string sDBUserID;
2412 string sDBPrescribedDrug;
2413 string sDBDisease;
2414 string sDBPrescriber;
2415 #
2416 numeric nDBDrugRelatedProblemField;
2417 numeric nDBRXTypeField;
2418 numeric nDBPatientRiskField;
2419 numeric nDBProblemTypeField;
2420 numeric nDBRecommendationField;
2421 numeric nDBActivityField;
2422 numeric nDBResultField;
2423 numeric nDBTimeField;
2424 #
2425 string sDBPatientIDField;
2426 string sDBAllergiesField;
2427 string sDBMedicalHistoryField;
2428 string sDBFamilyHistoryField;
2429 string sDBSocialHistoryField;
2430 #
2431 string sDBSOAPSubjectiveField;
2432 string sDBSOAPObjectiveField;
2433 string sDBSOAPAssessmentField;
2434 string sDBSOAPPlanField;
2435 string sDBSOAPFollowUpField;
2436 #
2437 string sDBLabValues;
2438 numeric nDBDOIScore;
```

```
2439
2440 #
2441 #-----
2442 # db fields for PIDS XML database...
2443 #-----
2444 #
2445 numeric nDBXMLPIDSKeyField;
2446 numeric nDBXMLSynced;
2447 #
2448 string sDBXMLIntervention;
2449 end;
2450
2451 #
2452 #-----
2453 # DATABASE OBJECT DEFINITIONS
2454 #-----
2455 #
2456 dbfile dbPids;
2457
2458 field nDBPIDSKeyField; # This is the record "key" index
2459 field nDBSynced;
2460 #
2461 field sDBRotationID;
2462 field sDBLocationID;
2463 field sDBInterventionDate;
2464 field sDBUserID;
2465 field sDBPrescribedDrug;
2466 field sDBDisease;
2467 field sDBPrescriber;
2468 #
2469 field nDBDrugRelatedProblemField;
2470 field nDBRXTypeField;
2471 field nDBPatientRiskField;
2472 field nDBProblemTypeField;
2473 field nDBRecommendationField;
2474 field nDBActivityField;
2475 field nDBResultField;
2476 field nDBTimeField;
2477 #
2478 field sDBPatientIDField;
2479 field sDBAllergiesField;
2480 field sDBMedicalHistoryField;
2481 field sDBFamilyHistoryField;
2482 field sDBSocialHistoryField;
2483 #
2484 field sDBSOAPSubjectiveField;
2485 field sDBSOAPObjectiveField;
2486 field sDBSOAPAssessmentField;
2487 field sDBSOAPPlanField;
2488 field sDBSOAPFollowUpField;
2489 #
2490 field sDBLabValues;
2491 #
2492 field nDBDOIScore;
2493
2494 sync_pref pda_to_pc;
2495
2496 end;
2497
2498 dbfile dbPidsXML;
2499
2500 field nDBXMLPIDSKeyField; # This is the record "key" index
2501 field nDBXMLSynced;
2502 field sDBXMLIntervention;
2503
2504 sync_pref pda_to_pc;
2505
2506 end;
2507
2508 dbfile dbPidsPrefs;
2509
```

```
2510     field nDBPrefsKeyField;
2511     field fDBPrefsEnableSpeedEntryField;
2512     field sDBPrefsWirelessURLField;
2513     field sDBPrefsLocationIDField;
2514     field sDBPrefsRotationIDField;
2515     field sDBPrefsUserIDField;
2516
2517     sync_pref pda_to_pc;
2518
2519 end;
2520
2521
```

```
2522 #
2523 # PIDS_error_handling.cpk
2524 #
2525 #####
2526 #
2527 # PIDS (tm)
2528 # Pharmacy Intervention Documentation System
2529 # Copyright © 2002 Integrated Software Systems, L.L.C.
2530 #
2531 #####
2532 #
2533 # Version: 1.0
2534 # Date: 12/23/2001
2535 # Author: Todd Viegut & Brad Tice
2536 #
2537 #
2538 # Function for displaying a standardized message box for PIDS errors.
2539 #
2540 function ErrorMsg(string sMessage);
2541     variables;
2542     numeric nResult;
2543     end;
2544     nResult = message_box(1, "PIDS Error", sMessage + " [" + ce_geterror + "]", "", "Ok",
2545     "");
2546 end;
2547
```

```
2548 #
2549 # PIDS_misc.cpk
2550 #
2551 #####
2552 #
2553 # PIDS (tm)
2554 # Pharmacy Intervention Documentation System
2555 # Copyright © 2002 Integrated Software Systems, L.L.C.
2556 #
2557 #####
2558 #
2559 # Version: 1.0
2560 # Date: 12/14/2001
2561 # Author: Todd Viegut & Brad Tice
2562 #
2563 #
2564 # Miscellaneous functions.
2565 #
2566 #
2567 #
2568 #-----
2569 # Automated data entry functions...
2570 #-----
2571 #
2572 function GetDRPTypeIndex(numeric nDrugRelatedProblemSelection) as numeric;
2573     variables;
2574         numeric fContinue;
2575         numeric nIndex;
2576     end;
2577     nIndex = 1;
2578     #
2579     fContinue = true;
2580     while ((nIndex < nTotalDRPFrames) and (fContinue));
2581         if ((nDrugRelatedProblemSelection >= nDRPSelectionListOffsets[nIndex]) and
2582             (nDrugRelatedProblemSelection < nDRPSelectionListOffsets[nIndex + 1]));
2583             fContinue = false;
2584         else;
2585             nIndex = nIndex + 1;
2586         end_if;
2587     end_while;
2588     #
2589     GetDRPTypeIndex = nIndex;
2590 end;
2591
2592 function GetDRPTypeSubIndex(numeric nDrugRelatedProblemSelection) as numeric;
2593     variables;
2594         numeric nIndex;
2595     end;
2596     nIndex = GetDRPTypeIndex(nDrugRelatedProblemSelection);
2597     GetDRPTypeSubIndex = nDrugRelatedProblemSelection - nDRPSelectionListOffsets[nIndex];
2598 end;
2599
2600 function GetDrugRelatedProblemData(numeric nDrugRelatedProblemSelection) as string;
2601     variables;
2602         object objSelector;
2603         numeric nIndex;
2604         numeric nSubIndex;
2605     end;
2606     nIndex = GetDRPTypeIndex(nDrugRelatedProblemSelection);
2607     nSubIndex = GetDRPTypeSubIndex(nDrugRelatedProblemSelection);
2608     objSelector = objDRPSelectors[nIndex - 1]; # This array is 0 based...
2609     #
2610     GetDrugRelatedProblemData = "DRUG RELATED PROBLEM:" + char(10) +
2611     drug_related_problem_list[nIndex] + " - " + objSelector.display;
2612 end;
2613
2613 function GetPatientInfoData() as string;
2614     GetPatientInfoData = ""; # Must initialize to empty string...
2615     #
2616     # Allergies:
```

```
2617     if (length(txAllergies.display) > 0);
2618         GetPatientInfoData = "ALLERGIES:" + char(10);
2619         GetPatientInfoData = GetPatientInfoData + txAllergies.display;
2620     end_if;
2621     #
2622     # Medication History:
2623     if (length(txMedHistory.display) > 0);
2624         if (length(GetPatientInfoData) > 0);
2625             GetPatientInfoData = GetPatientInfoData + char(10) + char(10) + "MED HISTORY:"
                + char(10);
2626         else;
2627             GetPatientInfoData = "MED HISTORY:" + char(10);
2628         end_if;
2629         GetPatientInfoData = GetPatientInfoData + txMedHistory.display;
2630     end_if;
2631     #
2632     # Family History:
2633     if (length(txFamilyHistory.display) > 0);
2634         if (length(GetPatientInfoData) > 0);
2635             GetPatientInfoData = GetPatientInfoData + char(10) + char(10) + "FAMILY
                HISTORY:" + char(10);
2636         else;
2637             GetPatientInfoData = "FAMILY HISTORY:" + char(10);
2638         end_if;
2639         GetPatientInfoData = GetPatientInfoData + txFamilyHistory.display;
2640     end_if;
2641     #
2642     # Social History:
2643     if (length(txSocialHistory.display) > 0);
2644         if (length(GetPatientInfoData) > 0);
2645             GetPatientInfoData = GetPatientInfoData + char(10) + char(10) + "SOCIAL
                HISTORY:" + char(10);
2646         else;
2647             GetPatientInfoData = "SOCIAL HISTORY:" + char(10);
2648         end_if;
2649         GetPatientInfoData = GetPatientInfoData + txSocialHistory.display;
2650     end_if;
2651 end;
2652
2653 function GetLabValuesData(string sInitialString) as string;
2654     variables;
2655         numeric nLoopIndex;
2656     end;
2657     GetLabValuesData = sInitialString;
2658     for nLoopIndex = 0, nLoopIndex < nLabValuesCreated;
2659         if (length(GetLabValuesData) > 0);
2660             GetLabValuesData = GetLabValuesData + char(10) +
                completed_lab_values[nLoopIndex];
2661         else;
2662             GetLabValuesData = GetLabValuesData + completed_lab_values[nLoopIndex];
2663         end_if;
2664     next nLoopIndex + 1;
2665 end;
2666
2667 #
2668 #-----
2669 # DOI Score(tm) calculation...
2670 #-----
2671 #
2672 function CalculatedDOIScore as numeric;
2673     variables;
2674         object objSelector;
2675         numeric nFrameIndex;
2676         numeric nWeightIndex;
2677         numeric nSelectedIndex;
2678     end;
2679     #
2680     CalculateDOIScore = 0;
2681     #
2682     for nFrameIndex = 2, nFrameIndex <= 9;
2683         objSelector = objInterventionSelectors[nFrameIndex];
```

```

2684     nSelectedIndex = objSelector.selected;
2685     #
2686     if nFrameIndex = 2;
2687         objSelector = objDRPSelectors[nDRPSelected - 1];
2688         nSelectedIndex = objSelector.selected;
2689         nWeightIndex = (nDRPSelectionListOffsets[nDRPSelected] + nSelectedIndex);
2690         CalculateDOIScore = CalculateDOIScore + nDRPWeights[nWeightIndex];
2691     else if nFrameIndex = 3;
2692         CalculateDOIScore = CalculateDOIScore + nRXTypeWeights[nSelectedIndex];
2693     else if nFrameIndex = 4;
2694         CalculateDOIScore = CalculateDOIScore + nPatientRiskWeights[nSelectedIndex];
2695     else if nFrameIndex = 5;
2696         CalculateDOIScore = CalculateDOIScore + nProblemTypeWeights[nSelectedIndex];
2697     else if nFrameIndex = 6;
2698         CalculateDOIScore = CalculateDOIScore +
2699             nRecommendationWeights[nSelectedIndex];
2700     else if nFrameIndex = 7;
2701         CalculateDOIScore = CalculateDOIScore + nActivityWeights[nSelectedIndex];
2702     else if nFrameIndex = 8;
2703         CalculateDOIScore = CalculateDOIScore + nResultWeights[nSelectedIndex];
2704     else if nFrameIndex = 9;
2705         CalculateDOIScore = CalculateDOIScore + nTimeWeights[nSelectedIndex];
2706     end if;
2707     next nFrameIndex + 1;
2708 end;
2709 #
2710 #-----
2711 # Functions invoked for frame navigation using
2712 # BACK and NEXT buttons...
2713 #-----
2714 #
2715 function HideFrame;
2716     if (nCurrentFrame >= 0) and (nCurrentFrame < nTotalFrames);
2717         if (nContainsSelector[nCurrentFrame]);
2718             hide objInterventionSelectors[nCurrentFrame];
2719         end if;
2720         hide objPIDSFrames[nCurrentFrame];
2721     end if;
2722 end;
2723
2724 function ShowFrame;
2725     if ((nCurrentFrame >= 0) and (nCurrentFrame < nTotalFrames));
2726         if objPIDSFrames[nCurrentFrame] = save_intervention_frame;
2727             txDOIScore.display = string(CalculateDOIScore, "#");
2728         end if;
2729         if (nContainsSelector[nCurrentFrame]);
2730             show objInterventionSelectors[nCurrentFrame];
2731         end if;
2732         show objPIDSFrames[nCurrentFrame];
2733     end if;
2734 end;
2735
2736 #
2737 #-----
2738 # Initialization function used at start up of
2739 # the PIDS application...
2740 #-----
2741 #
2742 function Init;
2743     nLabValuesCreated = 0;
2744     #
2745     text_entry_allergies_selector.object_string = sTitles[0];
2746     text_entry_med_history_selector.object_string = sTitles[1];
2747     text_entry_family_history_selector.object_string = sTitles[2];
2748     text_entry_social_history_selector.object_string = sTitles[3];
2749     #
2750     drug_selector.list = drugs_list;
2751     disease_selector.list = diseases_list;
2752     text_entry_allergies_selector.list = allergies_list;
2753     text_entry_med_history_selector.list = drugs_list;

```



```

2754     text_entry_family_history_selector.list = diseases_list;
2755     text_entry_social_history_selector.list = social_list;
2756     #
2757     nDBPrefsKeyField = 1;
2758     fDBPrefsEnableSpeedEntryField = false;
2759     sDBPrefsWirelessURLField = "";
2760     sDBPrefsLocationIDField = "";
2761     sDBPrefsRotationIDField = "";
2762     sDBPrefsUserIDField = "";
2763     #
2764     if (dba_ffOpenDB(dbPidsPrefs, "dcPidsPrefs"));
2765         if (dba_ffRecordExists(dbPidsPrefs, 1));
2766             if (dba_ffReadExactRecord(dbPidsPrefs, 1) = false);
2767                 ErrorMsg("Unable to read from the PIDS preferences database.");
2768             end_if;
2769         else if (dba_ffWriteRecord(dbPidsPrefs, 1) = false);
2770             ErrorMsg("Unable to write to the PIDS preferences database.");
2771         end_if;
2772         call dba_CloseDB(dbPidsPrefs);
2773     else;
2774         ErrorMsg("Unable to open the PIDS preferences database.");
2775     end_if;
2776     #
2777     fSpeedEntryEnabled = fDBPrefsEnableSpeedEntryField;
2778     sWirelessURL = sDBPrefsWirelessURLField;
2779     sLocationID = sDBPrefsLocationIDField;
2780     sRotationID = sDBPrefsRotationIDField;
2781     sUserID = sDBPrefsUserIDField;
2782     #
2783 end;
2784
2785 #
2786 #-----
2787 # Function used to retrieve the next key id...
2788 #-----
2789 #
2790 function GetInterventionKeyId as numeric;
2791     if (fUpdateMode);
2792         GetInterventionKeyId = nInterventionKeyId;
2793     else;
2794         GetInterventionKeyId = -1;
2795         if (dba_ffOpenDB(dbPids, "dbPids"));
2796             #
2797             # Retrieve the total records currently in the database...
2798             errorcode = ce_no_error;
2799             seek_end dbPids, 1;
2800             if (errorcode = ce_no_error);
2801                 errorcode = ce_no_error;
2802                 get_fields dbPids;
2803                 if (errorcode = ce_no_error);
2804                     GetInterventionKeyId = nDBPIDSKeyField + 1;
2805                 else if (errorcode = ce_eof);
2806                     GetInterventionKeyId = 1;
2807                 else;
2808                     ErrorMsg("Unable to obtain the next key id from the PIDS database.");
2809                 end_if;
2810             else;
2811                 ErrorMsg("Unable to obtain the next key id from the PIDS database.");
2812             end_if;
2813             #
2814             dba_CloseDB(dbPids);
2815         else;
2816             ErrorMsg("Unable to open PIDS preferences database to obtain the next key
2817             id.");
2818         end_if;
2819     end_if;
2820 end;
2821 #
2822 #-----
2823 # DB manipulation functions.

```

```

2824 #-----
2825 #
2826 function SaveXMLInterventionRecord(numeric nKey, string sSyncURL) as numeric;
2827     SaveXMLInterventionRecord = false;
2828     if (dba_ffOpenDB(dbPidsXML, "dbPidsXML"));
2829         #
2830         # Now save the XML record...
2831         nDBXMLPIDSKeyField = nDBPIDSKeyField;
2832         nDBXMLSynced = 0; # Always must be 0 (not synced)
2833         sDBXMLIntervention = BuildXMLRecord(sSyncURL);
2834         #
2835         SaveXMLInterventionRecord = dba_ffWriteRecord(dbPidsXML, nKey);
2836         if (not SaveXMLInterventionRecord);
2837             ErrorMsg("Unable to save the intervention to the PIDS XML database.");
2838         end_if;
2839         #
2840         dba_CloseDB(dbPidsXML);
2841     else;
2842         ErrorMsg("Unable to open PIDS interventions XML database.");
2843     end_if;
2844 end;
2845
2846 function SaveInterventionRecord(numeric nKey, string sSyncURL) as numeric;
2847     variables;
2848     numeric nResult;
2849 end;
2850 SaveInterventionRecord = false;
2851 if (dba_ffOpenDB(dbPids, "dbPids"));
2852     SaveInterventionRecord = dba_ffWriteRecord(dbPids, nKey);
2853     if (not SaveInterventionRecord);
2854         dba_CloseDB(dbPids);
2855         ErrorMsg("Unable to save the intervention to the PIDS database.");
2856     else;
2857         dba_CloseDB(dbPids);
2858         #
2859         # Save was successful so proceed to save the corresponding
2860         # XML record...
2861         SaveInterventionRecord = SaveXMLInterventionRecord(nKey, sSyncURL);
2862         if (not SaveInterventionRecord);
2863             #
2864             # Since we now require both databases to have identical records
2865             # (keys, etc.) we must "back out" the original saved intervention
2866             # since the XML formatted record was not successfully saved.
2867             nResult = dba_ffRemoveRecord(dbPids, nKey);
2868         end_if;
2869     end_if;
2870 else;
2871     ErrorMsg("Unable to open PIDS interventions database.");
2872 end_if;
2873 end;
2874
2875 function DeleteXMLInterventionRecord(numeric nKey) as numeric;
2876     DeleteXMLInterventionRecord = false;
2877     if (dba_ffOpenDB(dbPidsXML, "dbPidsXML"));
2878         DeleteXMLInterventionRecord = dba_ffRemoveRecord(dbPidsXML, nKey);
2879         if (not (DeleteXMLInterventionRecord));
2880             ErrorMsg("Unable to delete the intervention.");
2881         end_if;
2882         #
2883         dba_CloseDB(dbPidsXML);
2884     else;
2885         ErrorMsg("Unable to open PIDS interventions XML database.");
2886     end_if;
2887 end;
2888
2889 function DeleteInterventionRecord(numeric nKey) as numeric;
2890     DeleteInterventionRecord = false;
2891     if (dba_ffOpenDB(dbPids, "dbPids"));
2892         DeleteInterventionRecord = dba_ffRemoveRecord(dbPids, nKey);
2893         if (not (DeleteInterventionRecord));

```

```

2895         dba_CloseDB(dbPids);
2896         ErrorMsg("Unable to delete the intervention.");
2897     else;
2898         dba_CloseDB(dbPids);
2899         # Delete successful so proceed to delete the corresponding
2900         # XML record...
2901         DeleteInterventionRecord = DeleteXMLInterventionRecord(nKey);
2902     end_if;
2903
2904     else;
2905         ErrorMsg("Unable to open PIDS interventions database.");
2906     end_if;
2907 end;
2908
2909 function SavePreferences as numeric;
2910     SavePreferences = false;
2911     if (dba_ffOpenDB(dbPidsPrefs, "dbPidsPrefs"));
2912         #
2913         nDBPrefsKeyField = 1;
2914         fDBPrefsEnableSpeedEntryField = cbPrefsSpeedEntryCheckbox.checkbox;
2915         sDBPrefsWirelessURLField = txPrefsWirelessSyncURL.display;
2916         sDBPrefsLocationIDField = txPrefsLocationID.display;
2917         sDBPrefsRotationIDField = txPrefsRotationID.display;
2918         sDBPrefsUserIDField = txPrefsUserID.display;
2919         #
2920         SavePreferences = dba_ffWriteRecord(dbPidsPrefs, nDBPrefsKeyField);
2921         if (SavePreferences);
2922             fSpeedEntryEnabled = fDBPrefsEnableSpeedEntryField;
2923             sWirelessURL = sDBPrefsWirelessURLField;
2924             sLocationID = sDBPrefsLocationIDField;
2925             sRotationID = sDBPrefsRotationIDField;
2926             sUserID = sDBPrefsUserIDField;
2927             #
2928             # NOTE may need to delete this code out later...
2929             txLocationID.display = sLocationID;
2930             txRotationID.display = sRotationID;
2931             txUserID.display = sLocationID;
2932         else;
2933             ErrorMsg("Unable to save to PIDS preferences database.");
2934         end_if;
2935         #
2936         dba_CloseDB(dbPidsPrefs);
2937     else;
2938         ErrorMsg("Unable to open PIDS preferences database.");
2939     end_if;
2940 end;
2941
2942 #
2943 #-----
2944 # Pop-up selector functions...
2945 #-----
2946 #
2947 function Compare(string sString1, string sString2) as numeric;
2948     variables;
2949         numeric nChar1;
2950         numeric nChar2;
2951         numeric nIndex;
2952         numeric nLength;
2953         numeric fFinished;
2954     end;
2955     #
2956     # -1 indicates that sString1 < sString2
2957     # 0 indicates that they are equal
2958     # 1 indicates that sString1 > sString2
2959     #
2960     if (length(sString1) < length(sString2));
2961         nLength = length(sString1);
2962     else_if (length(sString1) > length(sString2));
2963         nLength = length(sString2);
2964     else;
2965         nLength = length(sString1);

```

```

2966     end_if;
2967     #
2968     fFinished = false;
2969     nIndex = 0;
2970     Compare = 0; # Always start by assuming they're equal...
2971     while ((nIndex < nLength) and (not fFinished));
2972         nChar1 = asc(mid(sString1, nIndex, 1));
2973         nChar2 = asc(mid(sString2, nIndex, 1));
2974         if (nChar1 < nChar2);
2975             Compare = -1;
2976             fFinished = true;
2977         else_if (nChar1 > nChar2);
2978             Compare = 1;
2979             fFinished = true;
2980         else;
2981             nIndex = nIndex + 1;
2982         end_if;
2983     end_while;
2984     #
2985     if (Compare = 0);
2986         if (length(sString1) < length(sString2));
2987             Compare = -1;
2988         else;
2989             Compare = 1;
2990         end_if;
2991     end_if;
2992 end;
2993
2994 function BinarySearch(string saList[], string sItem, numeric nFirst, numeric nLast) as
numeric;
2995     variables;
2996         numeric nMid;
2997     end;
2998     if (nFirst > nLast);
2999         BinarySearch = -1;
3000     else;
3001         nMid = Trunc((nFirst + nLast) / 2);
3002         if (sItem = saList[nMid]);
3003             BinarySearch = nMid;
3004         else_if (Compare(sItem, saList[nMid]) = -1); # sItem < saList[nMid]...
3005             BinarySearch = BinarySearch(saList, sItem, nFirst, (nMid - 1));
3006         else;
3007             BinarySearch = BinarySearch(saList, sItem, (nMid + 1), nLast);
3008         end_if;
3009     end_if;
3010 end;
3011
3012 function BestFitSearch(string saList[], string sItem, numeric nFirst, numeric nLast) as
numeric;
3013     variables;
3014         numeric nMid;
3015     end;
3016     if (nFirst > nLast);
3017         BestFitSearch = nFirst; # Only difference between binary and best fit is this
line...
3018     else;
3019         nMid = Trunc((nFirst + nLast) / 2);
3020         if (sItem = saList[nMid]);
3021             BestFitSearch = nMid;
3022         else_if (Compare(sItem, saList[nMid]) = -1); # sItem < saList[nMid]...
3023             BestFitSearch = BestFitSearch(saList, sItem, nFirst, (nMid - 1));
3024         else;
3025             BestFitSearch = BestFitSearch(saList, sItem, (nMid + 1), nLast);
3026         end_if;
3027     end_if;
3028 end;
3029
3030 function ShowDrugSelector;
3031     variables;
3032         string sBestFitString;
3033         numeric nSelectedIndex;

```

```
3034     end;
3035     if (length(txPrescribedDrug.display) > 0);
3036         sBestFitString = txPrescribedDrug.display;
3037         if (length(sBestFitString) >= 1);
3038             sBestFitString = upper(left(sBestFitString, 1)) + mid(sBestFitString, 1,
3039                 (length(sBestFitString) - 1));
3040         end_if;
3041         nSelectedIndex = BestFitSearch(drugs_list, sBestFitString, 0, (nDrugsListSize -
3042             1));
3043         drug_selector.selected = nSelectedIndex;
3044     end_if;
3045     show drug_selector;
3046 end;
3047
3048 function DrugSelected;
3049     variables;
3050         string sItemSelected;
3051     end;
3052     hide drug_selector;
3053     get invoker, sItemSelected;
3054     txPrescribedDrug.display = sItemSelected;
3055 end;
3056
3057 function ShowDiseaseSelector;
3058     variables;
3059         string sBestFitString;
3060         numeric nSelectedIndex;
3061     end;
3062     if (length(txDisease.display) > 0);
3063         # Must start at element 2 since item 0 and 1 are "Unknown" and "Other" which
3064         # aren't in alphabetical order and would make this BestFirstSearch() call fail...
3065         sBestFitString = txDisease.display;
3066         if (length(sBestFitString) >= 1);
3067             sBestFitString = upper(left(sBestFitString, 1)) + mid(sBestFitString, 1,
3068                 (length(sBestFitString) - 1));
3069         end_if;
3070         nSelectedIndex = BestFitSearch(diseases_list, sBestFitString, 2,
3071             (nDiseasesListSize - 1));
3072         disease_selector.selected = nSelectedIndex;
3073     end_if;
3074     show disease_selector;
3075 end;
3076
3077 function DiseaseSelected;
3078     variables;
3079         string sItemSelected;
3080     end;
3081     hide disease_selector;
3082     get invoker, sItemSelected;
3083     txDisease.display = sItemSelected;
3084     if (sItemSelected = "Asthma");
3085         call HideFrame;
3086         show DSW_asthma_frame;
3087     end_if;
3088 end;
3089
3090 function ShowTextEntrySelector;
3091     variables;
3092         string sSelectorType;
3093     end;
3094     sSelectorType = invoker.object_string;
3095     if (sSelectorType = sTitles[0]);
3096         show text_entry_allergies_selector;
3097     else_if (sSelectorType = sTitles[1]);
3098         show text_entry_med_history_selector;
3099     else_if (sSelectorType = sTitles[2]);
3100         show text_entry_family_history_selector;
3101     else_if (sSelectorType = sTitles[3]);
3102         show text_entry_social_history_selector;
3103     end_if;
3104 end;
```

```
3101
3102 function ShowPatientIDSelector;
3103     show patient_id_selector;
3104 end;
3105
3106 function GenericItemSelected;
3107     variables;
3108     string sSelectorType;
3109     string sItemSelected;
3110 end;
3111 sSelectorType = invoker.object_string;
3112 if (sSelectorType = sTitles[0]);
3113     hide text_entry_allergies_selector;
3114 else_if (sSelectorType = sTitles[1]);
3115     hide text_entry_med_history_selector;
3116 else_if (sSelectorType = sTitles[2]);
3117     hide text_entry_family_history_selector;
3118 else_if (sSelectorType = sTitles[3]);
3119     hide text_entry_social_history_selector;
3120 end_if;
3121 #
3122 get invoker, sItemSelected;
3123 if (length(txEntry.display) > 0);
3124     txEntry.display = txEntry.display + char(10) + sItemSelected;
3125 else;
3126     txEntry.display = sItemSelected;
3127 end_if;
3128 end;
3129
3130 #
3131 #-----
3132 # Lab Values functions...
3133 #-----
3134 #
3135 function LabValueTypeSelected;
3136     variables;
3137     numeric nSelectedItem;
3138 end;
3139 lab_value_types_label.display = invoker.display;
3140 #
3141 nSelectedItem = invoker.selected;
3142 # Blood Pressure
3143 if (nSelectedItem = 1);
3144     type_descriptions_selector.list = blood_pressure_list;
3145 # Respiratory
3146 else_if (nSelectedItem = 2);
3147     type_descriptions_selector.list = respiratory_value_list;
3148 # Blood Sugar
3149 else_if (nSelectedItem = 3);
3150     type_descriptions_selector.list = blood_sugar_values;
3151 # Lipids
3152 else_if (nSelectedItem = 4);
3153     type_descriptions_selector.list = lipid_values_list;
3154 # Osteoporosis
3155 else_if (nSelectedItem = 5);
3156     type_descriptions_selector.list = osteoporosis_values_list;
3157 # Liver Function Tests
3158 else_if (nSelectedItem = 6);
3159     type_descriptions_selector.list = liver_function_tests_list;
3160 # Electrolytes
3161 else_if (nSelectedItem = 7);
3162     type_descriptions_selector.list = electrolytes_list;
3163 # Anticoagulation
3164 else_if (nSelectedItem = 8);
3165     type_descriptions_selector.list = anticoagulation_values_list;
3166 # Renal Function
3167 else_if (nSelectedItem = 9);
3168     type_descriptions_selector.list = renal_function_list;
3169 # Hematology
3170 else_if (nSelectedItem = 10);
3171     type_descriptions_selector.list = hematology_list;
```

```
3172     end_if;
3173     #
3174     type_descriptions_selector.selected = 0;
3175     type_descriptions_label.display = type_descriptions_selector.display;
3176 end;
3177
3178 function TypeDescriptionsSelected;
3179     txtLabValue.focused = 1; # Set the focus on the value text field...
3180     type_descriptions_label.display = invoker.display;
3181 end;
3182
3183 function UnitsOfMeasureSelected;
3184     units_of_measure_label.display = invoker.display;
3185 end;
3186
3187 function ShowLabValueTypesSelector;
3188     show lab_value_types_selector;
3189 end;
3190
3191 function ShowTypeDescriptionsSelector;
3192     show type_descriptions_selector;
3193 end;
3194
3195 function ShowUnitsOfMeasureSelector;
3196     show units_of_measure_selector;
3197 end;
3198
3199 function EnterLabValues;
3200     call HideFrame;
3201     show add_lab_values_frame;
3202     lab_value_types_selector.selected = 0;
3203     lab_value_types_label.display = lab_value_types_selector.display;
3204     type_descriptions_selector.selected = 0;
3205     type_descriptions_label.display = type_descriptions_selector.display;
3206     units_of_measure_selector.selected = 0;
3207     units_of_measure_label.display = units_of_measure_selector.display;
3208     completed_lab_values_selector.selected = -1;
3209 end;
3210
3211 function AddLabValue;
3212     variables;
3213         string sUnitsOfMeasure;
3214         string sLabValueDescription;
3215     end;
3216     #
3217     if ((nLabValuesCreated < 20) and (lab_value_types_selector.selected > 0) and
        (type_descriptions_selector.selected > 0) and (length(txtLabValue.display) > 0));
3218         if (units_of_measure_selector.selected > 0);
3219             sUnitsOfMeasure = " " + units_of_measure_label.display;
3220         else;
3221             sUnitsOfMeasure = "";
3222         end_if;
3223         sLabValueDescription = type_descriptions_label.display + " " + txtLabValue.display
        + sUnitsOfMeasure;
3224         completed_lab_values[nLabValuesCreated] = sLabValueDescription;
3225         #
3226         nLabValuesCreated = nLabValuesCreated + 1;
3227         #
3228         lab_value_types_selector.selected = 0;
3229         lab_value_types_label.display = lab_value_types_selector.display;
3230         #
3231         type_descriptions_selector.selected = 0;
3232         type_descriptions_label.display = type_descriptions_selector.display;
3233         #
3234         units_of_measure_selector.selected = 0;
3235         units_of_measure_label.display = units_of_measure_selector.display;
3236         #
3237         completed_lab_values_selector.selected = -1;
3238         #
3239         show completed_lab_values_selector;
3240         #
```

```
3241     txtLabValue.display = "";
3242     end_if;
3243 end;
3244
3245 function ClearLabValuesList;
3246     variables;
3247     numeric nIndex;
3248     end;
3249     for nIndex = 0, nIndex < 20;
3250         completed_lab_values[nIndex] = "";
3251     next nIndex + 1;
3252     nLabValuesCreated = 0;
3253 end;
3254
3255 function AcceptLabValues;
3256     hide add_lab_values_frame;
3257     call ShowFrame;
3258     txtLabValues.display = GetLabValuesData(txtLabValues.display);
3259     txObjective.display = GetLabValuesData(txObjective.display);
3260     call ClearLabValuesList;
3261 end;
3262
3263 function CancelLabValues;
3264     hide add_lab_values_frame;
3265     call ShowFrame;
3266 end;
3267
3268 function DeleteLabValue;
3269     variables;
3270     numeric nIndex;
3271     numeric nLoopIndex;
3272     end;
3273     #
3274     nIndex = completed_lab_values_selector.selected;
3275     if ((nLabValuesCreated > 0) and (nIndex < nLabValuesCreated));
3276         nLabValuesCreated = nLabValuesCreated - 1;
3277         #
3278         for nLoopIndex = nIndex, nLoopIndex < nLabValuesCreated;
3279             completed_lab_values[nLoopIndex] = completed_lab_values[nIndex + 1];
3280         next nLoopIndex + 1;
3281         #
3282         for nLoopIndex = nLabValuesCreated, nLoopIndex < 20;
3283             completed_lab_values[nLoopIndex] = "";
3284         next nLoopIndex + 1;
3285         #
3286         completed_lab_values_selector.selected = -1;
3287         #
3288         show completed_lab_values_selector;
3289     else;
3290         completed_lab_values_selector.selected = -1;
3291     end_if;
3292 end;
3293
3294 function PickUserSelectedInterventionDate;
3295     variables;
3296     string sPreviousDate;
3297     end;
3298     sPreviousDate = intervention_date_button.display;
3299     intervention_date_button.display = GetInterventionDate(sPreviousDate);
3300 end;
3301
3302
```



```

3303 #
3304 # PIDS_sync.cpk
3305 #
3306 #####
3307 #
3308 # PIDS (tm)
3309 # Pharmacy Intervention Documentation System
3310 # Copyright © 2002 Integrated Software Systems, L.L.C.
3311 #
3312 #####
3313 #
3314 # Version: 1.0
3315 # Date: 11/24/2001
3316 # Author: Todd Viegut & Brad Tice
3317 #
3318 #
3319 # Suite of functions to perform wireless synchronization..
3320 #
3321 include "PIDS_error_handling.cpk";
3322 include "PIDS_db_files.cpk";
3323 include "dba_DBAAccess.cpk"; # include generic db access functions
3324 #
3325 #
3326 variables;
3327     string sInputString;
3328 end;
3329 #
3330 #-----
3331 # SOCKET CONNECTION DEFINITIONS
3332 # FOR INTERNET ACCESS
3333 #-----
3334 #
3335 #
3336 file sync_socket; # file object for triggering open socket
3337 end;
3338 #
3339 #
3340 # Read data from the socket
3341 function sync_socket;
3342     variables;
3343         numeric selection;
3344     end;
3345     errorcode = 0;
3346     get sync_socket, sInputString;
3347     if errorcode = 0;
3348         if bytesread <> 0;
3349             #txResponse.display = txResponse.display + sInputString;
3350         end_if;
3351     else;
3352         selection = message_box(1,"PIDS Error", "Wireless sync error occurred during read:
3353             " + cne_geterror, "", "Ok", "");
3354     end_if;
3354 end;
3355 #
3356 variables;
3357     string DONT_ENCODE =
3358         "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZC123456789_-.*";
3359     string HEX_DIGITS = "C123456789ABCDEF";
3360 end;
3361 function Hex(numeric nDecimal) as string;
3362     variables;
3363         numeric nQuotient;
3364         numeric nRemainder;
3365     end;
3366     nQuotient = trunc(nDecimal / 16);
3367     nRemainder = nDecimal % 16;
3368     #
3369     if (nQuotient > 0);
3370         Hex = Hex(nQuotient) + mid(HEX_DIGITS, nRemainder, 1);
3371     else;

```

```

3372     Hex = mid(HEX_DIGITS, nRemainder, 1);
3373 end_if;
3374 end;
3375
3376 function URLEncode(string sInput) as string;
3377     variables;
3378     string sHex;
3379     string sOutput;
3380     string sChar;
3381     numeric nLoop;
3382 end;
3383 for nLoop = 0, nLoop < length(sInput);
3384     sChar = mid(sInput, nLoop, 1);
3385     #
3386     # If the current char is not in the DONT_ENCODE
3387     # string then proceed to encode it...
3388     if (not (find(sChar, DONT_ENCODE, 0) >= 0));
3389         if (sChar = " ");
3390             sOutput = sOutput + "+";
3391         else;
3392             sHex = Hex(asc(sChar));
3393             if (length(sHex) = 1) or (length(sHex) = 3);
3394                 sHex = "0" + sHex;
3395             end_if;
3396             if (length(sHex) = 4);
3397                 sOutput = sOutput + "%" + left(sHex, 2);
3398                 sOutput = sOutput + "%" + right(sHex, 2);
3399             else;
3400                 sOutput = sOutput + "%" + sHex;
3401             end_if;
3402         end_if;
3403     else;
3404         sOutput = sOutput + sChar;
3405     end_if;
3406 next nLoop + 1;
3407 URLEncode = sOutput;
3408 end;
3409
3410 function BuildPostRequest(string sRelSyncURL, string sURLEncodedPostString) as string;
3411     BuildPostRequest = "POST " + sRelSyncURL + " HTTP/1.0" + char(13) + char(10)
3412         + "Content-type: application/x-www-form-urlencoded" + char(13) +
3413         char(10)
3414         + "Content-length: " + string(length(sURLEncodedPostString), "#") +
3415         char(13) + char(10) + char(13) + char(10)
3416         + sURLEncodedPostString
3417         + char(13) + char(10);
3418 end;
3419
3420 function BuildURLEncodedPostString() as string;
3421     variables;
3422     string sPostString;
3423 end;
3424 #
3425 sPostString = sPostString + "pids_1=" + string(nDBPIDSKeyField, "#") + "&";
3426 sPostString = sPostString + "pids_2=" + string(nDBSynced, "#") + "&";
3427 #
3428 sPostString = sPostString + "pids_3=" + URLEncode(sDBRotationID) + "&";
3429 sPostString = sPostString + "pids_4=" + URLEncode(sDBLocationID) + "&";
3430 sPostString = sPostString + "pids_5=" + URLEncode(sDBInterventionDate) + "&";
3431 sPostString = sPostString + "pids_6=" + URLEncode(sDBUserID) + "&";
3432 sPostString = sPostString + "pids_7=" + URLEncode(sDBPrescribedDrug) + "&";
3433 sPostString = sPostString + "pids_8=" + URLEncode(sDBDisease) + "&";
3434 sPostString = sPostString + "pids_9=" + URLEncode(sDBPrescriber) + "&";
3435 #
3436 sPostString = sPostString + "pids_10=" + string(nDBDrugRelatedProblemField, "#") +
3437 "&";
3438 sPostString = sPostString + "pids_11=" + string(nDBRXTypeField, "#") + "&";
3439 sPostString = sPostString + "pids_12=" + string(nDBPatientRiskField, "#") + "&";
3440 sPostString = sPostString + "pids_13=" + string(nDBProblemTypeField, "#") + "&";
3441 sPostString = sPostString + "pids_14=" + string(nDBRecommendationField, "#") + "&";
3442 sPostString = sPostString + "pids_15=" + string(nDBActivityField, "#") + "&";

```

```

3440 sPostString = sPostString + "pids_16=" + string(nDBResultField, "#") + "&";
3441 sPostString = sPostString + "pids_17=" + string(nDBTimeField, "#") + "&";
3442 #
3443 sPostString = sPostString + "pids_18=" + URLEncode(sDBPatientIDField) + "&";
3444 sPostString = sPostString + "pids_19=" + URLEncode(sDBAllergiesField) + "&";
3445 sPostString = sPostString + "pids_20=" + URLEncode(sDBMedicalHistoryField) + "&";
3446 sPostString = sPostString + "pids_21=" + URLEncode(sDBFamilyHistoryField) + "&";
3447 sPostString = sPostString + "pids_22=" + URLEncode(sDBSocialHistoryField) + "&";
3448 #
3449 sPostString = sPostString + "pids_23=" + URLEncode(sDBSOAPSSubjectiveField) + "&";
3450 sPostString = sPostString + "pids_24=" + URLEncode(sDBSOAPObjectiveField) + "&";
3451 sPostString = sPostString + "pids_25=" + URLEncode(sDBSOAPAssessmentField) + "&";
3452 sPostString = sPostString + "pids_26=" + URLEncode(sDBSOAPPlanField) + "&";
3453 sPostString = sPostString + "pids_27=" + URLEncode(sDBSOAPFollowUpField) + "&";
3454 #
3455 sPostString = sPostString + "pids_28=" + URLEncode(sDBLabValues) + "&";
3456 sPostString = sPostString + "pids_29=" + string(nDBDOIScore, "#");
3457 #
3458 BuildURLEncodedPostString = sPostString;
3459 end;
3460
3461 function BuildXMLRecord(string sSyncURL) as string;
3462     variables;
3463     string sXMLRecord;
3464 end;
3465 #
3466 sXMLRecord = sXMLRecord + "<?xml version='1.0'?>";
3467 sXMLRecord = sXMLRecord + "<!DOCTYPE intervention_set PUBLIC '-//Integrated Software  
Systems, LLC//Intervention Sets//EN'  
'http://www.pidsware.com/dtds/intervention_set_1_0.dtd'>";
3468 sXMLRecord = sXMLRecord + "<intervention_set sync_url='" + sSyncURL + "'>";
3469 sXMLRecord = sXMLRecord + "<intervention>";
3470 sXMLRecord = sXMLRecord + "<key url_param_name='pids_1'" + string(nDBPIDSKeyField, "#") + "</key>";
3471 sXMLRecord = sXMLRecord + "<sync_flag url_param_name='pids_2'" + string(nDBSynced, "#") + "</sync_flag>";
3472 #
3473 sXMLRecord = sXMLRecord + "<rotation_id url_param_name='pids_3'" + sDBRotationID + "</rotation_id>";
3474 sXMLRecord = sXMLRecord + "<location_id url_param_name='pids_4'" + sDBLocationID + "</location_id>";
3475 sXMLRecord = sXMLRecord + "<intervention_date url_param_name='pids_5'" + sDBInterventionDate + "</intervention_date>";
3476 sXMLRecord = sXMLRecord + "<user_id url_param_name='pids_6'" + sDBUserID + "</user_id>";
3477 sXMLRecord = sXMLRecord + "<prescribed_drug url_param_name='pids_7'" + sDBPrescribedDrug + "</prescribed_drug>";
3478 sXMLRecord = sXMLRecord + "<disease url_param_name='pids_8'" + sDBDisease + "</disease>";
3479 sXMLRecord = sXMLRecord + "<prescriber url_param_name='pids_9'" + sDBPrescriber + "</prescriber>";
3480 #
3481 sXMLRecord = sXMLRecord + "<drug_related_probleml url_param_name='pids_10'" + string(nDBDrugRelatedProblemField, "#") + "</drug_related_probleml>";
3482 sXMLRecord = sXMLRecord + "<rx_type url_param_name='pids_11'" + string(nDBRXTypeField, "#") + "</rx_type>";
3483 sXMLRecord = sXMLRecord + "<patient_risk url_param_name='pids_12'" + string(nDBPatientRiskField, "#") + "</patient_risk>";
3484 sXMLRecord = sXMLRecord + "<problem_type url_param_name='pids_13'" + string(nDBProblemTypeField, "#") + "</problem_type>";
3485 sXMLRecord = sXMLRecord + "<recommendation url_param_name='pids_14'" + string(nDBRecommendationField, "#") + "</recommendation>";
3486 sXMLRecord = sXMLRecord + "<activity url_param_name='pids_15'" + string(nDBActivityField, "#") + "</activity>";
3487 sXMLRecord = sXMLRecord + "<result url_param_name='pids_16'" + string(nDBResultField, "#") + "</result>";
3488 sXMLRecord = sXMLRecord + "<time url_param_name='pids_17'" + string(nDBTimeField, "#") + "</time>";
3489 #
3490 sXMLRecord = sXMLRecord + "<patient_id url_param_name='pids_18'" + sDBPatientIDField + "</patient_id>";

```

```

3491     sXMLRecord = sXMLRecord + "<allergies url_param_name='pids_19'>" + sDBAllergiesField +
3492     "</allergies>";
3493     sXMLRecord = sXMLRecord + "<medical_history url_param_name='pids_20'>" +
3494     sDBMedicalHistoryField + "</medical_history>";
3495     sXMLRecord = sXMLRecord + "<family_history url_param_name='pids_21'>" +
3496     sDBFamilyHistoryField + "</family_history>";
3497     sXMLRecord = sXMLRecord + "<social_history url_param_name='pids_22'>" +
3498     sDBSocialHistoryField + "</social_history>";
3499     #
3500     sXMLRecord = sXMLRecord + "<scap_subjective url_param_name='pids_23'>" +
3501     sDBSOAPSubjectiveField + "</soap_subjective>";
3502     sXMLRecord = sXMLRecord + "<scap_objective url_param_name='pids_24'>" +
3503     sDBSOAPObjectiveField + "</scap_objective>";
3504     sXMLRecord = sXMLRecord + "<soap_assessment url_param_name='pids_25'>" +
3505     sDBSOAPAssessmentField + "</soap_assessment>";
3506     sXMLRecord = sXMLRecord + "<scap_plan url_param_name='pids_26'>" + sDBSOAPPlanField +
3507     "</soap_plan>";
3508     sXMLRecord = sXMLRecord + "<scap_follow_up url_param_name='pids_27'>" +
3509     sDBSOAPFollowUpField + "</soap_follow_up>";
3510     #
3511     sXMLRecord = sXMLRecord + "<lab_values url_param_name='pids_28'>" + sDBLabValues +
3512     "</lab_values>";
3513     sXMLRecord = sXMLRecord + "<doi_score url_param_name='pids_29'>" + string(nDBDOIScore,
3514     "#") + "</doi_score>";
3515     sXMLRecord = sXMLRecord + "</intervention>";
3516     sXMLRecord = sXMLRecord + "</intervention_set>";
3517     #
3518     BuildXMLRecord = sXMLRecord;
3519 end;
3520 #
3521 # Write data to the socket
3522 function SyncIntervention(string sSyncURL, string sRequest) as numeric;
3523     close sync_socket;
3524     #
3525     open sync_socket, sSyncURL;
3526     sInputString = sRequest;
3527     #
3528     errorcode = ce_no_error;
3529     put sync_socket, sInputString, length(sInputString);
3530     #
3531     #close sync_socket;
3532     #
3533     SyncIntervention = errorcode;
3534 end;
3535 #
3536 function ExtractRelativeSyncURL(string sURL) as string;
3537     variables;
3538     numeric nIndex;
3539     end;
3540     ExtractRelativeSyncURL = "";
3541     nIndex = find("//", sURL, 0);
3542     if (nIndex > -1);
3543         nIndex = find("/", sURL, (nIndex + 2));
3544         ExtractRelativeSyncURL = Mid(sURL, nIndex, (length(sURL) - nIndex));
3545     end_if;
3546 end;
3547 #
3548 function SyncAllInterventions(string sSyncURL);
3549     variables;
3550     string sRelativeSyncURL;
3551     string URLEncodedString;
3552     string sRequest;
3553     numeric nLocalErrorCode;
3554     end;
3555     #
3556     # Retrieve the total records currently in the database...
3557     if (dba_ffOpenDB(dbPids, "dbPids"));
3558         sRelativeSyncURL = ExtractRelativeSyncURL(sSyncURL);
3559         errorcode = ce_no_error;
3560         seek_start dbPids, 0;

```

```
3551     while (errorcode = ce_no_error);
3552         get_fields dbPids;
3553         if errorcode = ce_no_error;
3554             sURLEncodedString = BuildURLEncodedPostString();
3555             sRequest = BuildPostRequest(sRelativeSyncURL, sURLEncodedString);
3556             nLocalErrorCode = SyncIntervention(sSyncURL, sRequest);
3557             errorcode = nLocalErrorCode;
3558             if errorcode <> ce_no_error;
3559                 ErrorMsg("Wireless sync error occurred.");
3560             end_if;
3561         end_if;
3562     end_while;
3563     call dba_CloseDB(dbPids);
3564 else;
3565     ErrorMsg("Unable to open PIDS intervention database.");
3566 end_if;
3567 end;
3568
3569
```

```
3570 #
3571 # PIDS_wizards.cpk
3572 #
3573 #####
3574 #
3575 # PIDS (tm)
3576 # Pharmacy Intervention Documentation System
3577 # Copyright © 2002 Integrated Software Systems, L.L.C.
3578 #
3579 #####
3580 #
3581 # Version: 1.0
3582 # Date: 1/10/2002
3583 # Author: Todd Viegut & Brad Tice
3584 #
3585 #
3586 # Disease State Wizards:
3587 #
3588 #
3589 #
3590 #
3591 #=====
3592 # A S T H M A D I S E A S E S T A T E W I Z A R D
3593 #=====
3594 #
3595 #
3596 #-----
3597 # Invoked functions associated with the
3598 # DSW Asthma Disclaimer screen...
3599 #-----
3600 #
3601 function DSWAcceptAsthmaDisclaimer;
3602     hide DSW_asthma_frame;
3603     show DSW_asthma_age_frame;
3604 end;
3605
3606 function DSWCancelAsthmaDisclaimer;
3607     hide DSW_asthma_frame;
3608     call ShowFrame;
3609 end;
3610
3611 #
3612 #-----
3613 # Invoked functions associated with the
3614 # DSW Asthma Age screen...
3615 #-----
3616 #
3617 function DSWBackToAsthmaDisclaimer;
3618     hide DSW_asthma_age_frame;
3619     show DSW_asthma_frame;
3620 end;
3621
3622 function DSWNextToAsthmaEval;
3623     hide DSW_asthma_age_frame;
3624     show DSW_asthma_eval_frame;
3625 end;
3626
3627 function DSWAsthmaAgeSelected;
3628 #     hide DSW_asthma_age_frame;
3629     if (invoker.selected > 0);
3630         show DSW_asthma_eval_frame;
3631         #
3632         days_with_symptoms_selector.selected = 0;
3633         nights_with_symptoms_selector.selected = 0;
3634         expiratory_flow_selector.selected = 0;
3635         expiratory_volume_selector.selected = 0;
3636         #
3637         days_with_symptoms_label.display = days_with_symptoms_selector.display;
3638         nights_with_symptoms_label.display = nights_with_symptoms_selector.display;
3639         expiratory_flow_label.display = expiratory_flow_selector.display;
3640         expiratory_volume_label.display = expiratory_volume_selector.display;
```

```
3641     end_if;
3642 end;
3643
3644 #
3645 #-----
3646 # Invoked functions associated with the
3647 # DSW Asthma Evaluation screen...
3648 #-----
3649 #
3650 function DSWSBackToAsthmaAge;
3651     hide DSW_asthma_eval_frame;
3652     show DSW_asthma_age_frame;
3653 end;
3654
3655 function DSWNextToAsthmaAge;
3656     hide DSW_asthma_eval_frame;
3657     show DSW_asthma_age_frame;
3658 end;
3659
3660 function DSWSHowDaysWithSymptomsSelector;
3661     show days_with_symptoms_selector;
3662 end;
3663
3664 function DSWSHowNightsWithSymptomsSelector;
3665     show nights_with_symptoms_selector;
3666 end;
3667
3668 function DSWSHowExpiratoryFlowSelector;
3669     show expiratory_flow_selector;
3670 end;
3671
3672 function DSWSHowExpiratoryVolumeSelector;
3673     show expiratory_volume_selector;
3674 end;
3675
3676 function DSWDaysWithSymptomsSelected;
3677     if (invoker.selected > 0);
3678         days_with_symptoms_label.display = invoker.display;
3679     end_if;
3680 end;
3681
3682 function DSWNightsWithSymptomsSelected;
3683     if (invoker.selected > 0);
3684         nights_with_symptoms_label.display = invoker.display;
3685     end_if;
3686 end;
3687
3688 function DSWExpiratoryFlowSelected;
3689     if (invoker.selected > 0);
3690         expiratory_flow_label.display = invoker.display;
3691     end_if;
3692 end;
3693
3694 function DSWExpiratoryVolumeSelected;
3695     if (invoker.selected > 0);
3696         expiratory_volume_label.display = invoker.display;
3697     end_if;
3698 end;
3699 #
3700 #=====
3701 #  L I P I D S   D I S E A S E   S T A T E   W I Z A R D
3702 #=====
3703 #
3704 #
3705
3706
```

```
1 package com.pidsware.pids.conduit;
2
3 import palm.conduit.*;
4 import java.io.*;
5 import java.net.URLEncoder;
6 /**
7  *
8  * InterventionRecord.java
9  *
10 * #####
11 *
12 * PIDS (tm)
13 * Pharmacy Intervention Documentation System
14 * PIDS (tm) Conduit for performing Palm HotSync®s
15 * Copyright © 2002 Integrated Software Systems, L.L.C.
16 *
17 * #####
18 *
19 *
20 * Subclassed record object for an intervention record on the HH.
21 *
22 *
23 * @version 1.0
24 * @date 08.10.2001
25 * @author: Todd Viegut & Brad Tice
26 */
27 public class InterventionRecord extends AbstractRecord {
28     private double interventionKey;
29     private double synced;
30     private String rotationID = null;
31     private String locationID = null;
32     private String interventionDate = null;
33     private String userID = null;
34     private String prescribedDrug = null;
35     private String disease = null;
36     private String prescriber = null;
37     private double drugRelatedProblem;
38     private double rxType;
39     private double patientRisk;
40     private double problemType;
41     private double recommendation;
42     private double activity;
43     private double result;
44     private double interventionDuration;
45     private String patientID = null;
46     private String allergies = null;
47     private String medicalHistory = null;
48     private String familyHistory = null;
49     private String socialHistory = null;
50     private String subjective = null;
51     private String objective = null;
52     private String assessment = null;
53     private String plan = null;
54     private String followUp = null;
55     private String labValues = null;
56     private double doiScore;
57     public double getActivity() {
58         return this.activity;
59     }
60     public String getAllergies() {
61         return this.allergies;
62     }
63     public String getAssessment() {
64         return this.assessment;
65     }
66     public String getDisease() {
67         return this.disease;
68     }
69     public double getDOIScore() {
70         return doiScore;
71     }
72 }
```



```
72 public double getDrugRelatedProblem() {
73     return this.drugRelatedProblem;
74 }
75 public String getFamilyHistory() {
76     return this.familyHistory;
77 }
78 public String getFollowUp() {
79     return this.followUp;
80 }
81 public String getInterventionDate() {
82     return this.interventionDate;
83 }
84 public double getInterventionDuration() {
85     return this.interventionDuration;
86 }
87 public double getInterventionKey() {
88     return this.interventionKey;
89 }
90 public String getLabValues() {
91     return this.labValues;
92 }
93 public String getLocationID() {
94     return this.locationID;
95 }
96 public String getMedicalHistory() {
97     return this.medicalHistory;
98 }
99 public String getObjective() {
100     return this.objective;
101 }
102 public String getPatientID() {
103     return this.patientID;
104 }
105 public double getPatientRisk() {
106     return this.patientRisk;
107 }
108 public String getPlan() {
109     return this.plan;
110 }
111 public String getPrescribedDrug() {
112     return this.prescribedDrug;
113 }
114 public String getPrescriber() {
115     return this.prescriber;
116 }
117 public double getProblemType() {
118     return this.problemType;
119 }
120 public double getRecommendation() {
121     return this.recommendation;
122 }
123 public double getResult() {
124     return this.result;
125 }
126 /**
127  * Insert the method's description here.
128  * Creation date: (12/29/2001 11:25:36 AM)
129  * @return java.lang.String
130  */
131 public String getRotationID() {
132     return this.rotationID;
133 }
134 public double getRXType() {
135     return this.rxType;
136 }
137 public String getSocialHistory() {
138     return this.socialHistory;
139 }
140 public String getSubjective() {
141     return this.subjective;
142 }
```

```
143 public double getSynced() {
144     return this.synced;
145 }
146 public String getUserID() {
147     return this.userID;
148 }
149 /** Returns a null String if the parameter s is empty (contains no characters).
150 */
151 String nullIfEmpty(String s) {
152     if ("".equals(s)) {
153         return null;
154     } else {
155         return s;
156     }
157 }
158 /** Converts a DataInputStream representing the body of the hand-held device
159 * record into Record object members. The format of the DataInputStream
160 * must be understood and translated by the developer.
161 */
162 public void readData(DataInputStream in) throws IOException {
163     //
164     //=====
165     //      I N T E R V E N T I O N   H E A D E R
166     //=====
167     //
168     // PIDS intervention key
169     this.interventionKey = in.readDouble();
170
171     // Lab Values
172     this.synced = in.readDouble();
173
174     // Rotation ID
175     this.rotationID = readCString(in);
176
177     // Site ID
178     this.locationID = readCString(in);
179
180     // Intervention Date
181     this.interventionDate = readCString(in);
182
183     // Student ID
184     this.userID = readCString(in);
185
186     // Prescribed Drug
187     this.prescribedDrug = readCString(in);
188
189     // Disease
190     this.disease = readCString(in);
191
192     // Prescriber
193     this.prescriber = readCString(in);
194
195     //
196     //=====
197     //      P A T I E N T   I N F O
198     //=====
199     //
200     // Drug Related Problem
201     this.drugRelatedProblem = in.readDouble();
202
203     // RX Type
204     this.rxType = in.readDouble();
205
206     // Patient Risk
207     this.patientRisk = in.readDouble();
208
209
210
211
212
213
```

```
214 // Problem Type
215 this.problemType = in.readDouble();
216
217 // Recommendation
218 this.recommendation = in.readDouble();
219
220
221 // Activity
222 this.activity = in.readDouble();
223
224
225 // Result
226 this.result = in.readDouble();
227
228
229 // InterventionDuration
230 this.interventionDuration = in.readDouble();
231
232
233 //
234 //=====
235 //          P A T I E N T   I N F O
236 //=====
237 //
238
239 // Patient ID
240 this.patientID = readCString(in);
241
242 // Allergies notes
243 this.allergies = readCString(in);
244
245 // Medical History notes
246 this.medicalHistory = readCString(in);
247
248 // Family History notes
249 this.familyHistory = readCString(in);
250
251 // Social History notes
252 this.socialHistory = readCString(in);
253
254 //
255 //=====
256 //          S O A P   N O T E S
257 //=====
258 //
259
260 // Suojective notes
261 this.subjective = readCString(in);
262
263 // Objective notes
264 this.objective = readCString(in);
265
266 // Assessment notes
267 this.assessment = readCString(in);
268
269 // Plan notes
270 this.plan = readCString(in);
271
272 // Follow Up notes
273 this.followUp = readCString(in);
274
275 //
276 //=====
277 //          L A B   V A L U E S
278 //=====
279 //
280
281 // Lab Values
282 this.labValues = readCString(in);
283
284 //
```

```
285 //=====
286 //          D O I   S C O R E
287 //=====
288 //
289
290 // DOI Score
291 this.doiScore = in.readDouble();
292
293 //
294 //=====
295 //=====
296 //=====
297 //=====
298 //=====
299 //=====
300 //=====
301 //=====
302 //
303 }
304 public void setActivity(double value) {
305     this.activity = value;
306 }
307 public void setAllergies(String value) {
308     this.allergies = nullIfEmpty(value);
309 }
310 public void setAssessment(String value) {
311     this.assessment = nullIfEmpty(value);
312 }
313 public void setDisease(String value) {
314     this.disease = nullIfEmpty(value);
315 }
316 public void setDOIScore(double value) {
317     this.doiScore = value;
318 }
319 public void setDrugRelatedProblem(double value) {
320     this.drugRelatedProblem = value;
321 }
322 public void setFamilyHistory(String value) {
323     this.familyHistory = nullIfEmpty(value);
324 }
325 public void setFollowUp(String value) {
326     this.followUp = nullIfEmpty(value);
327 }
328 public void setInterventionDate(String value) {
329     this.interventionDate = nullIfEmpty(value);
330 }
331 public void setInterventionDuration(double value) {
332     this.interventionDuration = value;
333 }
334 public void setInterventionKey(double value) {
335     this.interventionKey = value;
336 }
337 public void setLabValues(String value) {
338     this.labValues = nullIfEmpty(value);
339 }
340 public void setLocationID(String value) {
341     this.locationID = nullIfEmpty(value);
342 }
343 public void setMedicalHistory(String value) {
344     this.medicalHistory = nullIfEmpty(value);
345 }
346 public void setObjective(String value) {
347     this.objective = nullIfEmpty(value);
348 }
349 public void setPatientID(String value) {
350     this.patientID = nullIfEmpty(value);
351 }
352 public void setPatientRisk(double value) {
353     this.patientRisk = value;
354 }
355 public void setPlan(String value) {
```

```

356     this.plan = nullIfEmpty(value);
357 }
358 public void setPrescribedDrug(String value) {
359     this.prescribedDrug = nullIfEmpty(value);
360 }
361 public void setPrescriber(String value) {
362     this.prescriber = nullIfEmpty(value);
363 }
364 public void setProblemType(double value) {
365     this.problemType = value;
366 }
367 public void setRecommendation(double value) {
368     this.recommendation = value;
369 }
370 public void setResult(double value) {
371     this.result = value;
372 }
373 public void setRotationID(String value) {
374     this.rotationID = nullIfEmpty(value);
375 }
376 public void setRXType(double value) {
377     this.rxType = value;
378 }
379 public void setSocialHistory(String value) {
380     this.socialHistory = nullIfEmpty(value);
381 }
382 public void setSubjective(String value) {
383     this.subjective = nullIfEmpty(value);
384 }
385 public void setSynced(double value) {
386     this.synced = value;
387 }
388 public void setUserID(String value) {
389     this.userID = nullIfEmpty(value);
390 }
391 static String stringWithoutCarriageReturns(String aString) {
392     StringBuffer sb = new StringBuffer();
393     int i, c;
394     char ch;
395
396     for (i = 0, c = aString.length(); i < c; i++) {
397         ch = aString.charAt(i);
398         if (ch == '\r' && (i + 1) < c && aString.charAt(i + 1) == '\n')
399             continue;
400         else
401             sb.append(ch);
402     }
403     return sb.toString();
404 }
405 public String toFormattedString() {
406     return ("Intervention record: {\r\n" +
407         "    key: " + getInterventionKey() + "\r\n" +
408         "    synced: " + getSynced() + "\r\n" +
409         "    rotation ID: " + getRotationID() + "\r\n" +
410         "    site ID: " + getLocationID() + "\r\n" +
411         "    intervention date: " + getInterventionDate() + "\r\n" +
412         "    student ID: " + getUserID() + "\r\n" +
413         "    prescribed drug: " + getPrescribedDrug() + "\r\n" +
414         "    disease: " + getDisease() + "\r\n" +
415         "    prescriber: " + getPrescriber() + "\r\n" +
416         "    drug related problem: " + getDrugRelatedProblem() + "\r\n" +
417         "    RX type: " + getRXType() + "\r\n" +
418         "    patient risk: " + getPatientRisk() + "\r\n" +
419         "    problem type: " + getProblemType() + "\r\n" +
420         "    recommendation: " + getRecommendation() + "\r\n" +
421         "    activity: " + getActivity() + "\r\n" +
422         "    result: " + getResult() + "\r\n" +
423         "    intervention duration: " + getInterventionDuration() + "\r\n" +
424
425         "    patient ID: " + getPatientID() + "\r\n" +
426         "    allergies: " + getAllergies() + "\r\n" +

```

```

426         " medical history: " + getMedicalHistory() + "\r\n" +
427         " family history: " + getFamilyHistory() + "\r\n" +
428         " social history: " + getSocialHistory() + "\r\n" +
429         " SOAP NOTES: " + "\r\n" +
430         "----- + "\r\n" +
431         " subjective: " + getSubjective() + "\r\n" +
432         " objective: " + getObjective() + "\r\n" +
433         " assessment: " + getAssessment() + "\r\n" +
434         " plan: " + getPlan() + "\r\n" +
435         " follow up: " + getFollowUp() + "\r\n\r\n" +
436         "----- + "\r\n" +
437         " lab values: " + getLabValues() + "\r\n" +
438         " DOI score: " + getDOIScore() + "\r\n" +
439         "}\r\n" +
440         super.toFormattedString());
441     }
442     public String toPostString() {
443         StringBuffer postString = new StringBuffer();
444         //
445         postString
446             .append("pids_1=")
447             .append(URLEncoder.encode(Double.toString(getInterventionKey()))))
448             .append("&");
449         postString
450             .append("pids_2=")
451             .append(URLEncoder.encode(Double.toString(getSynced()))))
452             .append("&");
453         //
454         postString.append("pids_3=").append(URLEncoder.encode(getRotationID())).append(
455             "&");
456         postString.append("pids_4=").append(URLEncoder.encode(getLocationID())).append(
457             "&");
458         postString
459             .append("pids_5=")
460             .append(URLEncoder.encode(getInterventionDate()))
461             .append("&");
462         postString.append("pids_6=").append(URLEncoder.encode(getUserID())).append("&");
463         postString
464             .append("pids_7=")
465             .append(URLEncoder.encode(getPrescribedDrug()))
466             .append("&");
467         postString.append("pids_8=").append(URLEncoder.encode(getDisease())).append(
468             "&");
469         postString.append("pids_9=").append(URLEncoder.encode(getPrescriber())).append(
470             "&");
471         //
472         postString
473             .append("pids_10=")
474             .append(URLEncoder.encode(Double.toString(getDrugRelatedProblem()))))
475             .append("&");
476         postString
477             .append("pids_11=")
478             .append(URLEncoder.encode(Double.toString(getRXType()))))
479             .append("&");
480         postString
481             .append("pids_12=")
482             .append(URLEncoder.encode(Double.toString(getPatientRisk()))))
483             .append("&");
484         postString
485             .append("pids_13=")
486             .append(URLEncoder.encode(Double.toString(getProblemType()))))
487             .append("&");
488         postString
489             .append("pids_14=")
490             .append(URLEncoder.encode(Double.toString(getRecommendation()))))
491             .append("&");
492         postString
493             .append("pids_15=")
494             .append(URLEncoder.encode(Double.toString(getActivity()))))
495             .append("&");
496         postString

```

```

497         .append("pids_16=")
498         .append(URLEncoder.encode(Double.toString(getResult()))))
499         .append("&");
500     postString
501         .append("pids_17=")
502         .append(URLEncoder.encode(Double.toString(getInterventionDuration()))))
503         .append("&");
504     //
505     postString.append("pids_18=").append(URLEncoder.encode(getPatientID())).append(
506         "&");
507     postString.append("pids_19=").append(URLEncoder.encode(getAllergies())).append(
508         "&");
509     postString
510         .append("pids_20=")
511         .append(URLEncoder.encode(getMedicalHistory()))
512         .append("&");
513     postString
514         .append("pids_21=")
515         .append(URLEncoder.encode(getFamilyHistory()))
516         .append("&");
517     postString
518         .append("pids_22=")
519         .append(URLEncoder.encode(getSocialHistory()))
520         .append("&");
521     //
522     postString.append("pids_23=").append(
523         URLEncoder.encode(getSubjective())).append(
524         "&");
525     postString.append("pids_24=").append(URLEncoder.encode(getObjective())).append(
526         "&");
527     postString.append("pids_25=").append(
528         URLEncoder.encode(getAssessment())).append(
529         "&");
530     postString.append("pids_26=").append(URLEncoder.encode(getPlan())).append("&");
531     postString.append("pids_27=").append(URLEncoder.encode(getFollowUp())).append(
532         "&");
533     //
534     postString.append("pids_28=").append(URLEncoder.encode(getLabValues())).append(
535         "&");
536     postString.append("pids_29=").append(
537         URLEncoder.encode(Double.toString(getDOIScore())));
538     //
539     return postString.toString();
540 }
541 public String toString() {
542     return "Intervention record:  <rotation id="
543         + getRotationID()
544         + ">  <site id="
545         + getLocationID()
546         + ">  <patient id="
547         + getPatientID()
548         + ">  <prescribed drug="
549         + getPrescribedDrug()
550         + ">  "
551         + super.toString();
552 }
553 public void writeData(DataOutputStream out) throws IOException {
554     /*
555     //
556     //=====
557     //      I N T E R V E N T I O N   H E A D E R
558     //=====
559     //
560
561     // PIDS intervention key
562     out.writeDouble(this.interventionKey);
563
564     // Synced
565     out.writeDouble(this.synced);
566
567     // Rotation ID

```

```
568     writeCString(out, stringWithoutCarriageReturns(this.rotationID));
569
570     // Site ID
571     writeCString(out, stringWithoutCarriageReturns(this.locationID));
572
573     // Intervention Date
574     writeCString(out, stringWithoutCarriageReturns(this.interventionDate));
575
576     // Student ID
577     writeCString(out, stringWithoutCarriageReturns(this.userID));
578
579     // Prescribed Drug
580     writeCString(out, stringWithoutCarriageReturns(this.prescribedDrug));
581
582     // Disease
583     writeCString(out, stringWithoutCarriageReturns(this.disease));
584
585     // Prescriber
586     writeCString(out, stringWithoutCarriageReturns(this.prescriber));
587
588     //
589     //=====
590     //      I N T E R V E N T I O N      I N F O
591     //=====
592     //
593
594     // Drug Related Problem
595     out.writeDouble(this.drugRelatedProblem);
596
597     // RX Type
598     out.writeDouble(this.rxType);
599
600     // Patient Risk
601     out.writeDouble(this.patientRisk);
602
603     // Problem Type
604     out.writeDouble(this.problemType);
605
606     // Recommendation
607     out.writeDouble(this.recommendation);
608
609     // Activity
610     out.writeDouble(this.activity);
611
612     // Result
613     out.writeDouble(this.result);
614
615     // InterventionDuration
616     out.writeDouble(this.interventionDuration);
617
618     //
619     //=====
620     //      P A T I E N T      I N F O
621     //=====
622     //
623
624     // Patient ID
625     writeCString(out, stringWithoutCarriageReturns(this.patientID));
626
627     // Allergies notes
628     writeCString(out, stringWithoutCarriageReturns(this.allergies));
629
630     // Medical History notes
631     writeCString(out, stringWithoutCarriageReturns(this.medicalHistory));
632
633     // Family History notes
634     writeCString(out, stringWithoutCarriageReturns(this.familyHistory));
635
636     // Social History notes
637     writeCString(out, stringWithoutCarriageReturns(this.socialHistory));
638
```



```
639    //
640    //=====
641    //          S O A P   N O T E S
642    //=====
643    //
644
645    // Subjective notes
646    writeCString(out, stringWithoutCarriageReturns(this.subjective));
647
648    // Objective notes
649    writeCString(out, stringWithoutCarriageReturns(this.objective));
650
651    // Assessment notes
652    writeCString(out, stringWithoutCarriageReturns(this.assessment));
653
654    // Plan notes
655    writeCString(out, stringWithoutCarriageReturns(this.plan));
656
657    // Follow Up notes
658    writeCString(out, stringWithoutCarriageReturns(this.followUp));
659
660    //
661    //=====
662    //          L A B   V A L U E S
663    //=====
664    //
665
666    // Lab Values
667    writeCString(out, stringWithoutCarriageReturns(this.labValues));
668
669    //
670    //=====
671    //          D O I   S C O R E
672    //=====
673    //
674
675    // DOI Score
676    out.writeDouble(this.doiScore);
677
678    */
679
680    //
681    //=====
682    //=====
683    //=====
684    //=====
685    //=====
686    //=====
687    //=====
688    //=====
689    //
690
691    //
692    //=====
693    //          I N T E R V E N T I O N   H E A D E R
694    //=====
695    //
696
697    // PIDS intervention key
698    out.writeDouble(this.interventionKey);
699
700    // Synced
701    out.writeDouble(this.synced);
702
703    // Rotation ID
704    if (this.rotationID != null) {
705        out.write(this.rotationID.getBytes());
706        out.write(0);
707    }
708
709    // Site ID
```

```
710     if (this.locationID != null) {
711         out.write(this.locationID.getBytes());
712         out.write(0);
713     }
714
715     // Intervention Date
716     if (this.interventionDate != null) {
717         out.write(this.interventionDate.getBytes());
718         out.write(0);
719     }
720
721     // Student ID
722     if (this.userID != null) {
723         out.write(this.userID.getBytes());
724         out.write(0);
725     }
726
727     // Prescribed Drug
728     if (this.prescribedDrug != null) {
729         out.write(this.prescribedDrug.getBytes());
730         out.write(0);
731     }
732
733     // Disease
734     if (this.disease != null) {
735         out.write(this.disease.getBytes());
736         out.write(0);
737     }
738
739     // Prescriber
740     if (this.prescriber != null) {
741         out.write(this.prescriber.getBytes());
742         out.write(0);
743     }
744
745     //
746     //=====
747     //      I N T E R V E N T I O N   I N F O
748     //=====
749     //
750
751     // Drug Related Problem
752     out.writeDouble(this.drugRelatedProblem);
753
754     // RX Type
755     out.writeDouble(this.rxType);
756
757     // Patient Risk
758     out.writeDouble(this.patientRisk);
759
760     // Problem Type
761     out.writeDouble(this.problemType);
762
763     // Recommendation
764     out.writeDouble(this.recommendation);
765
766     // Activity
767     out.writeDouble(this.activity);
768
769     // Result
770     out.writeDouble(this.result);
771
772     // InterventionDuration
773     out.writeDouble(this.interventionDuration);
774
775     //
776     //=====
777     //      P A T I E N T   I N F O
778     //=====
779     //
780
```

```
781 // Patient ID
782 if (this.patientID != null) {
783     out.write(this.patientID.getBytes());
784     out.write(0);
785 }
786
787 // Allergies notes
788 if (this.allergies != null) {
789     out.write(this.allergies.getBytes());
790     out.write(0);
791 }
792
793 // Medical History notes
794 if (this.medicalHistory != null) {
795     out.write(this.medicalHistory.getBytes());
796     out.write(0);
797 }
798
799 // Family History notes
800 if (this.familyHistory != null) {
801     out.write(this.familyHistory.getBytes());
802     out.write(0);
803 }
804
805 // Social History notes
806 if (this.socialHistory != null) {
807     out.write(this.socialHistory.getBytes());
808     out.write(0);
809 }
810
811 //
812 //=====
813 //                S O A P   N O T E S
814 //=====
815 //
816
817 // Subjective notes
818 if (this.subjective != null) {
819     out.write(this.subjective.getBytes());
820     out.write(0);
821 }
822
823 // Objective notes
824 if (this.objective != null) {
825     out.write(this.objective.getBytes());
826     out.write(0);
827 }
828
829 // Assessment notes
830 if (this.assessment != null) {
831     out.write(this.assessment.getBytes());
832     out.write(0);
833 }
834
835 // Plan notes
836 if (this.plan != null) {
837     out.write(this.plan.getBytes());
838     out.write(0);
839 }
840
841 // Follow Up notes
842 if (this.followUp != null) {
843     out.write(this.followUp.getBytes());
844     out.write(0);
845 }
846
847 //
848 //=====
849 //                L A B   V A L U E S
850 //=====
851 //
```

```

852
853     // Lab Values
854
855     if (this.labValues != null) {
856         out.write(this.labValues.getBytes());
857         out.write(0);
858     }
859
860     //
861     //=====
862     //          D O I   S C O R E
863     //=====
864     //
865
866     // DOI Score
867     out.writeDouble(this.doiScore);
868 }
869
870
871
872
873
874
875
876
877
878
879
880 package com.pidsware.pids.conduit;
881
882 import java.util.*;
883
884 /**
885  *
886  * PIDSConduitBundle.java
887  *
888  *#####
889  *
890  * PIDS (tm)
891  * Pharmacy Intervention Documentation System
892  * PIDS (tm) Conduit for performing Palm HotSync®
893  * Copyright © 2002 Integrated Software Systems, L.L.C.
894  *
895  *#####
896  *
897  *
898  * Class providing localization for PIDS Conduit resources.
899  *
900  *
901  * @version 1.0
902  * @date    08.10.2001
903  * @author: Todd Viegut & Brad Tice
904  */
905
906 public class PIDSConduitBundle extends ListResourceBundle {
907     //
908     public static final char COPYRIGHT = ((char) 169);
909     public static final char REGISTERED = ((char) 174);
910     public static final char TM = ((char) 174); // This need to be changed to the actual
          char value...
911     //
912     static final Object[][] contents =
913     {
914         {"CONDUIT_NAME", "PIDS" + TM + " Conduit"},
915         {"pids_conduit_props", "pids.props"},
916         {"BAD_URL_ERROR", "PIDS" + TM + " Conduit Error: Bad URL: "},
917         {"SYNC_ERROR", "PIDS" + TM + " Conduit Error: Synchronization failed"},
918         {"RW_ERROR", "PIDS" + TM + " Conduit Error: Failed to successfully sync the data
          to the URL specified"},
919         {"FILE_NOT_FOUND_ERROR", "PIDS" + TM + " Conduit Error: URL Not Found: "},
920         {"OK_PIDS_CONDUIT", "OK PIDS" + TM + " Conduit"},

```

```

921         {"PIDS_CONDUIT", "PIDS" + TM + " Conduit"},
922         {"DIR_DELIMITER", "\\\"},
923         {"PIDS_Conduit_Do_Nothing", "PIDS" + TM + " Conduit - sync set to Do Nothing"},
924         {"debug", "true"},
925         {"logfile", "PIDScond.log"}
926     };
927     public Object[][] getContents() {
928         return contents;
929     }
930 }
931
932
933
934
935
936
937
938
939
940
941 package com.pidsware.pids.conduit;
942
943 import java.net.URL;
944 import java.net.HttpURLConnection;
945 import java.net.URLConnection;
946 import java.net.MalformedURLException;
947 import java.io.InputStreamReader;
948 import java.io.IOException;
949 import java.io.BufferedReader;
950 import java.io.CharArrayReader;
951 import java.io.Reader;
952
953 import org.xml.sax.*;
954 import com.ibm.xml.parsers.NonValidatingDOMParser;
955
956 /**
957  *
958  * DOMInterventionURLBuilder.java
959  *
960  * #####
961  *
962  * PIDS (tm)
963  * Pharmacy Intervention Documentation System
964  * PIDS (tm) Conduit for performing Palm HotSync's
965  * Copyright © 2002 Integrated Software Systems, L.L.C.
966  *
967  * #####
968  *
969  *
970  * This class is a non-validating DOM parser which takes in an
971  * XML formatted intervention, parses it, and dynamically builds
972  * a URL encoded name/value parameters list that gets sent as a
973  * POST request to the designated sync URL provided by the user
974  * on the HH.
975  *
976  *
977  * @version 1.0
978  * @date 02.04.2002
979  * @author: Todd Viegut & Brad Tice
980  */
981
982 import java.io.OutputStreamWriter;
983 import java.io.PrintWriter;
984 import java.io.UnsupportedEncodingException;
985
986 import org.w3c.dom.Attr;
987 import org.w3c.dom.Document;
988 import org.w3c.dom.NamedNodeMap;
989 import org.w3c.dom.Node;
990 import org.w3c.dom.NodeList;
991 import org.w3c.dom.Element;

```

```

992
993 public class DOMInterventionURLBuilder {
994     /** Default parser name. */
995     private static final String DEFAULT_PARSER_NAME =
996         "com.ibm.xml.parsers.NonValidatingDOMParser";
997
998     /** Buffer for URL building. */
999     private StringBuffer urlPostString = new StringBuffer();
1000
1001     private String syncURL;
1002
1003     private NonValidatingDOMParser parser;
1004     public DOMInterventionURLBuilder() throws ClassNotFoundException, InstantiationException,
1005     IllegalAccessException {
1006         this(DEFAULT_PARSER_NAME);
1007     }
1008     public DOMInterventionURLBuilder(String parserName) throws ClassNotFoundException,
1009     InstantiationException, IllegalAccessException {
1010         //
1011         this.parser = (NonValidatingDOMParser) Class.forName(parserName).newInstance();
1012     }
1013     public static void main(String argv[]) {
1014         String parserName = DEFAULT_PARSER_NAME;
1015         //
1016         try {
1017             DOMInterventionURLBuilder urlBuilder =
1018                 new DOMInterventionURLBuilder(parserName);
1019             //
1020             urlBuilder.parse("intervention_set.xml");
1021             //
1022             System.out.println("Redirect URL: " + urlBuilder.retrieveSyncURL());
1023             System.out.println("Post String: " + urlBuilder.retrievePostString());
1024             System.out.println(
1025                 urlBuilder.testPostRequest(
1026                     urlBuilder.retrievePostString(),
1027                     urlBuilder.retrieveSyncURL()));
1028             //
1029             urlBuilder.parse("intervention_set_1_0.xml");
1030             //
1031             System.out.println("Redirect URL: " + urlBuilder.retrieveSyncURL());
1032             System.out.println("Post String: " + urlBuilder.retrievePostString());
1033             System.out.println(
1034                 urlBuilder.testPostRequest(
1035                     urlBuilder.retrievePostString(),
1036                     urlBuilder.retrieveSyncURL()));
1037         } catch (Exception e) {
1038             e.printStackTrace();
1039         }
1040     }
1041     public void parse(Reader reader) {
1042         try {
1043             parser.reset();
1044             //
1045             this.syncURL = null;
1046             this.urlPostString = new StringBuffer();
1047             //
1048             InputSource inputSource = new InputSource(reader);
1049             //
1050             parser.parse(inputSource);
1051             Document document = parser.getDocument();
1052             //
1053             this.traverse(document);
1054         } catch (Exception e) {
1055             String message = null;
1056             char[] charArray = new char[50000];
1057             try {
1058                 int size = reader.read(charArray);
1059                 message = new String(charArray, 0, size);
1060             } catch (IOException ioException) {
1061                 message =
1062                     new String("An error occured. Unable to retrieve the XML record.\n\n");
1063             }
1064         }
1065     }

```

```

1060     }
1061     processError(message, e);
1062 }
1063 }
1064 public void parse(String uri) {
1065     try {
1066         parser.reset();
1067         //
1068         this.syncURL = null;
1069         this.urlPostString = new StringBuffer();
1070         //
1071         parser.parse(uri);
1072         Document document = parser.getDocument();
1073         //
1074         this.traverse(document);
1075     } catch (Exception e) {
1076         processError(uri, e);
1077     }
1078 }
1079 private static void printUsage() {
1080     System.err.println(
1081         "usage: com.pidsware.pids.conduit.DOMInterventionURLBuilder {options} uri ...";
1082     System.err.println();
1083     System.err.println("options:");
1084     System.err.println("  -d name    Specify DOM parser wrapper by name.");
1085     System.err.println("  -p parser  Default parser: " + DEFAULT_PARSER_NAME);
1086     System.err.println("  -h        This help screen.");
1087 }
1088 private void processError(String message, Throwable exception) {
1089     //
1090     EmailSender sender = new EmailSender(message, exception);
1091     sender.start();
1092     //
1093     exception.printStackTrace(System.err);
1094 }
1095 public String retrievePostString() {
1096     return this.urlPostString.toString();
1097 }
1098 public String retrieveSyncURL() {
1099     String syncURL = null;
1100     //
1101     Document interventionDoc = this.parser.getDocument();
1102     Element interventionSetRoot = interventionDoc.getDocumentElement();
1103     syncURL = interventionSetRoot.getAttribute("sync_url");
1104     //
1105     if ((syncURL == null) || (!syncURL.equals(syncURL))) {
1106         NodeList list = interventionSetRoot.getElementsByTagName("sync_url");
1107         Node node = list.item(0);
1108         if (node != null) {
1109             syncURL =
1110                 ((node.getFirstChild() != null)
1111                  && (node.getFirstChild().getNodeValue() != null))
1112                 ? node.getFirstChild().getNodeValue()
1113                 : "";
1114         }
1115     }
1116     return syncURL;
1117 }
1118 }
1119 private int testPostRequest(String postString, String syncURL) {
1120     int syncStatus = 0; // default is NOT successful...
1121     //
1122     HttpURLConnection connection = null;
1123     BufferedReader in = null;
1124     PrintWriter out = null;
1125     try {
1126         syncURL =
1127             (syncURL.toLowerCase().startsWith("tcp")
1128              ? "http" + syncURL.substring(syncURL.indexOf(": "))
1129              : syncURL);
1130         URL url = new URL(syncURL);

```

```

1131     connection = (URLConnection) url.openConnection();
1132     connection.setRequestMethod("POST");
1133     connection.setRequestProperty(
1134         "Content-Type",
1135         "application/x-www-form-urlencoded");
1136     connection.setRequestProperty(
1137         "Content-length",
1138         Integer.toString(postString.length()));
1139     connection.setDoOutput(true);
1140     connection.setDoInput(true);
1141     //
1142     out = new PrintWriter(connection.getOutputStream());
1143     out.write(postString);
1144     out.flush();
1145     //
1146     in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
1147     String inputLine;
1148     //
1149     if ((inputLine = in.readLine()) != null) {
1150         if ((inputLine != null) && (!"".equals(inputLine))) {
1151             syncStatus = Integer.parseInt(inputLine);
1152         }
1153     }
1154 } catch (MalformedURLException malformedURLException) {
1155     malformedURLException.printStackTrace();
1156 } catch (Exception e) {
1157     e.printStackTrace();
1158 } finally {
1159     try {
1160         if (in != null) {
1161             in.close();
1162         }
1163     } catch (IOException ioException) {
1164     }
1165     //
1166     if (out != null) {
1167         out.close();
1168     }
1169     //
1170     if (connection != null) {
1171         connection.disconnect();
1172     }
1173 }
1174 //
1175 return syncStatus;
1176 }
1177 private void traverse(Node node) {
1178     //
1179     // is there anything to do?
1180     if (node == null) {
1181         return;
1182     }
1183     //
1184     NodeList children = null;
1185     //
1186     int type = node.getNodeType();
1187     switch (type) {
1188         case Node.DOCUMENT_NODE :
1189             Node d = ((Document) node).getDocumentElement();
1190             traverse(d);
1191             break;
1192         case Node.ELEMENT_NODE :
1193             NamedNodeMap attrs = node.getAttributes();
1194             if (attrs != null) {
1195                 Node attrNode = attrs.getNamedItem("url_param_name");
1196                 if (attrNode != null) {
1197                     NodeList list = attrNode.getChildNodes();
1198                     this
1199                         .urlPostString
1200                         .append(((this.urlPostString.length() > 0) ? "&" : ""))
1201                         .append(attrNode.getNodeValue())

```



```

1202         .append("- ")
1203         .append(
1204             ((node.getFirstChild() != null)
1205              && (node.getFirstChild().getNodeValue() != null))
1206              ?
1207              java.net.URLEncoder.encode(node.getFirstChild().getNodeVal
1208              ue())
1209              : "");
1210     }
1211     //
1212     children = node.getChildNodes();
1213     if (children != null) {
1214         int len = children.getLength();
1215         for (int i = 0; i < len; i++) {
1216             traverse(children.item(i));
1217         }
1218         break;
1219     case Node.ENTITY_REFERENCE_NODE :
1220         children = node.getChildNodes();
1221         if (children != null) {
1222             int len = children.getLength();
1223             for (int i = 0; i < len; i++) {
1224                 traverse(children.item(i));
1225             }
1226         }
1227         break;
1228     case Node.CDATA_SECTION_NODE :
1229     case Node.TEXT_NODE :
1230         break;
1231     }
1232 }
1233 }
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244 package com.pidsware.pids.conduit;
1245
1246 import java.io.*;
1247 import sun.net.smtp.*;
1248 /**
1249  *
1250  * EmailSender.java
1251  *
1252  * #####
1253  *
1254  * PIDS (tm)
1255  * Pharmacy Intervention Documentation System
1256  * PIDS (tm) Conduit for performing Palm HotSync®s
1257  * Copyright © 2002 Integrated Software Systems, L.L.C.
1258  *
1259  * #####
1260  *
1261  *
1262  * The PIDS (tm) beta test auto error logging facility. This class
1263  * is spawned as a separate thread which will email any errors that
1264  * occur during a Palm HotSync® with PIDS (tm).
1265  *
1266  *
1267  * @version 1.0
1268  * @date 02.08.2002
1269  * @author: Todd Viegut & Brad Tice
1270  */

```

```
1271 public class EmailSender extends Thread {
1272     private String errorMsg;
1273     private Throwable exception;
1274     public EmailSender() {
1275         super();
1276     }
1277     public EmailSender(Runnable target) {
1278         super(target);
1279     }
1280     public EmailSender(Runnable target, String name) {
1281         super(target, name);
1282     }
1283     public EmailSender(String name) {
1284         super(name);
1285     }
1286     public EmailSender(String errorMsg, Throwable exception) {
1287         super();
1288         this.errorMsg = errorMsg;
1289         this.exception = exception;
1290     }
1291     public EmailSender(ThreadGroup group, Runnable target) {
1292         super(group, target);
1293     }
1294     public EmailSender(ThreadGroup group, Runnable target, String name) {
1295         super(group, target, name);
1296     }
1297     public EmailSender(ThreadGroup group, String name) {
1298         super(group, name);
1299     }
1300     public EmailSender(Throwable exception) {
1301         super();
1302         this.exception = exception;
1303     }
1304     public static void main(String[] args) {
1305         try {
1306             throw new java.io.IOException("TESTING");
1307         } catch (java.io.IOException e) {
1308             EmailSender sender = new EmailSender("OOOPS...", e);
1309             sender.start();
1310         }
1311     }
1312     /**
1313      * Creation date: (2/8/2002 10:32:21 AM)
1314      */
1315     public void run() {
1316         try {
1317             SmtplibClient smtpClient = new SmtplibClient("mail.pidsware.com");
1318
1319             //assumes localhost has the SMTP
1320             smtpClient.from("viegs@aol.com");
1321             smtpClient.to("conduit_errors@pidsware.com");
1322             PrintStream out = smtpClient.startMessage();
1323             out.println("To: conduit_errors@pidsware.com");
1324             out.println("From: viegs@aol.com");
1325             out.println("Subject: Conduit error...");
1326             if (this.errorMsg != null) {
1327                 out.println(this.errorMsg);
1328             }
1329             if (this.exception != null) {
1330                 exception.printStackTrace(out);
1331             }
1332             smtpClient.closeServer();
1333             //
1334             System.err.println("Conduit error email sent to beta support at
1335                               www.pidsware.com");
1336         } catch (Exception e) {
1337         }
1338     }
1339 }
1340
```

```
1341
1342
1343
1344
1345
1346
1347
1348
1349 package com.pidsware.pids.conduit;
1350
1351 import java.util.*;
1352 import java.net.*;
1353 import java.io.*;
1354 import palm.conduit.*;
1355 import org.xml.sax.*;
1356 import org.xml.sax.helpers.*;
1357
1358 /**
1359  *
1360  * PIDSConduit.java
1361  *
1362  * #####
1363  *
1364  * PIDS (tm)
1365  * Pharmacy Intervention Documentation System
1366  * PIDS (tm) Conduit for performing Palm HotSync@s
1367  * Copyright © 2002 Integrated Software Systems, L.L.C.
1368  *
1369  * #####
1370  *
1371  *
1372  * MAIN SOURCE CODE FILE FOR THE PIDS(tm) CONDUIT
1373  *
1374  *
1375  * @version 1.0
1376  * @date 08.10.2001
1377  * @author: Todd Viegut & Brad Tice
1378  */
1379 public class PIDSConduit implements Conduit {
1380     /** Reference to all strings used by PIDS Conduit */
1381     private static ResourceBundle pidsConduitBundle =
1382         ResourceBundle.getBundle("com.pidsware.pids.conduit.PIDSConduitBunale");
1383
1384     /** Name of the conduit - hardcoded since cannot currently get it from the registry */
1385     private static final String NAME = pidsConduitBundle.getString("CONDUIT_NAME");
1386
1387     /** This string should be replaced with the URL of the prc to be copied. */
1388     private static final String PROPS_FILE =
1389         pidsConduitBundle.getString("pids_conduit_props");
1390
1391     /** Vector of interventions to by synced. */
1392     private Vector interventions;
1393
1394     /** Intervention reference. */
1395     private InterventionRecord intervention;
1396
1397     /**
1398      * PIDSConduit constructor comment.
1399      */
1400     public PIDSConduit() {
1401         super();
1402     }
1403
1404     /**
1405      * An entry point to pass configuration parameters that is called by the
1406      * HotSync Manager conduit configuration.
1407      *
1408      * @param info ConfigureConduitInfo
1409      * @returns int
1410      */
1411     public int configure(ConfigureConduitInfo info) {
1412         ConduitConfigure config = new ConduitConfigure(info, NAME);
1413         config.createDialog();
1414     }
1415 }
```

```
1410
1411 // Only if the data has changed do we fill the info
1412 // parameter with the new values from the configuration dlg
1413 if (config.dataChanged) {
1414     //
1415     int propsValue = SyncProperties.SYNC_DO_NOTHING; // default
1416     propsValue = config.saveState;
1417
1418     if (config.setDefault) {
1419         //
1420         info.syncPermanent = propsValue;
1421         info.syncPref = ConfigureConduitInfo.PREF_PERMANENT;
1422     }
1423     info.syncNew = propsValue;
1424 }
1425 //
1426 Log.out("ConfigureConduitInfo: /n" + info.toString());
1427 //
1428 return 0;
1429 }
1430 /**
1431  * This method simply displays all intervention records for the database
1432  * passed in.
1433  *
1434  * @param dbName String The name of the database for which to display records.
1435  */
1436 private void displayInterventions(String dbName) throws SyncException, IOException {
1437     // Open our handheld DB
1438     int db = SyncManager.openDB(dbName, 0, SyncManager.OPEN_READ | SyncManager.OPEN_WRITE
1439         | SyncManager.OPEN_EXCLUSIVE);
1440
1441     // Find out how many intervention records there are in the database...
1442     int count = SyncManager.getDBRecordCount(db);
1443
1444     XMLInterventionRecord record = null;
1445     //
1446     // Loop over all records
1447     for (int i = 0; i < count; i++) {
1448         record = new XMLInterventionRecord();
1449         //
1450         record.setIndex(i);
1451         SyncManager.readRecordByIndex(db, record);
1452         //
1453         Log.out(record.toFormattedString());
1454     }
1455
1456     // Close DB
1457     SyncManager.closeDB(db);
1458 }
1459 /**
1460  * Starts the application.
1461  * @param args an array of command-line arguments
1462  */
1463 public static void main(String[] args) {
1464     // Insert code to start the application here.
1465     PIDSConduit conduit = new PIDSConduit();
1466 }
1467 /**
1468  * Returns a String representation of the conduit name.
1469  */
1470 public String name() {
1471     return NAME;
1472 }
1473 /**
1474  * Opens and runs a conduit. All conduit-specific functionality should be
1475  * entered in this method.
1476  * @param props SyncProperties
1477  */
1478 public void open(SyncProperties props) {
1479     //
```

```

1480     byte[] line = new byte[50000];
1481     //
1482     try {
1483         // check what type of sync is required
1484         if (props.syncType == SyncProperties.SYNC_DO_NOTHING) {
1485             //
1486             Log.out(pidsConduitBundle.getString("PIDS_Conduit_Do_Nothing"));
1487             Log.endSync();
1488             //
1489             return;
1490         }
1491
1492         //
1493         // FIRST, before we do any "syncing", we must purge all deleted interventions...
1494         purgeDeletedInterventions("dbPidsXML");
1495         purgeDeletedInterventions("dbPids");
1496
1497         //
1498         // NOW, we can proceed to sync the interventions.
1499         syncInterventions("dbPidsXML");
1500
1501         //
1502         Log.out(pidsConduitBundle.getString("OK_PIDS_CONDUIT"));
1503         Log.endSync(); // Log we are successful
1504     } catch (MalformedURLException ex) {
1505         processError(pidsConduitBundle.getString("BAD_URL_ERROR"), ex);
1506     } catch (SyncException e) {
1507         processError(pidsConduitBundle.getString("SYNC_ERROR"), e);
1508     } catch (IOException e) {
1509         processError(pidsConduitBundle.getString("RW_ERROR"), e);
1510     } catch (Exception e) {
1511         processError(pidsConduitBundle.getString("RW_ERROR"), e);
1512     }
1513 }
1514 /**
1515  * This method posts a request to synchronize a single intervention
1516  * record to the synchronization site passed in to it.
1517  *
1518  * @param postString String
1519  * @param syncURL String
1520  * @return int
1521  */
1522 private int postRequest(String postString, String syncURL) throws SyncException {
1523     int syncStatus = 0; // default is NOT successful...
1524     //
1525     HttpURLConnection connection = null;
1526     BufferedReader in = null;
1527     PrintWriter out = null;
1528     try {
1529         syncURL =
1530             (syncURL.toLowerCase().startsWith("tcp")
1531              ? "http" + syncURL.substring(syncURL.indexOf(": "))
1532              : syncURL);
1533         URL url = new URL(syncURL);
1534         connection = (HttpURLConnection) url.openConnection();
1535         connection.setRequestMethod("POST");
1536         connection.setRequestProperty(
1537             "Content-Type",
1538             "application/x-www-form-urlencoded");
1539         connection.setRequestProperty(
1540             "Content-length",
1541             Integer.toString(postString.length()));
1542         connection.setDoOutput(true);
1543         connection.setDoInput(true);
1544         //
1545         out = new PrintWriter(connection.getOutputStream());
1546         out.write(postString);
1547         out.flush();
1548         //
1549         in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
1550         String inputLine;

```

```

1551         //
1552         if ((inputLine = in.readLine()) != null) {
1553             if ((inputLine != null) && (!"".equals(inputLine))) {
1554                 syncStatus = Integer.parseInt(inputLine);
1555             }
1556         }
1557     } catch (Exception e) {
1558         e.printStackTrace();
1559     } finally {
1560         try {
1561             if (in != null) {
1562                 in.close();
1563             }
1564             catch (IOException ioException) {
1565             }
1566             //
1567             if (out != null) {
1568                 out.close();
1569             }
1570             //
1571             if (connection != null) {
1572                 connection.disconnect();
1573             }
1574         }
1575         //
1576         return syncStatus;
1577     }
1578 /**
1579  * This method processes any exception thrown during a HotSync. It also
1580  * launches the auto error logging thread which will send an email to
1581  * conduit_errors@pidsware.com.
1582  *
1583  * @param message String
1584  * @param exception Throwable
1585  */
1586 private void processError(String message, Throwable exception) {
1587     //
1588     // log(ex.toString());
1589     //
1590     EmailSender sender = new EmailSender(message, exception);
1591     sender.start();
1592     //
1593     exception.printStackTrace();
1594     //
1595     Log.err(message + " " + exception.toString() + "\n");
1596     Log.abortSync();
1597 }
1598 /**
1599  * This method will purge any deleted records from the HH for the database named passed to
1600  * it.
1601  *
1602  * @param dbName String
1603  */
1604 private void purgeDeletedInterventions(String dbName) throws SyncException {
1605     int db = SyncManager.openDB(dbName, 0, SyncManager.OPEN_READ | SyncManager.OPEN_WRITE
1606         | SyncManager.OPEN_EXCLUSIVE);
1607     SyncManager.purgeDeletedRecs(db);
1608     SyncManager.closeDB(db);
1609 }
1610 /**
1611  * This method will retrieve from the HH the url of the site to synchronize to.
1612  *
1613  * @return String
1614  * @param dbName String
1615  */
1616 private String readSyncURL(String dbName) throws SyncException, IOException {
1617     int db = -1;
1618     String url = null;
1619     try {
1620         //
1621         db =

```

```

1620         SyncManager.openDB(
1621             dbName,
1622             0,
1623             SyncManager.OPEN_READ | SyncManager.OPEN_EXCLUSIVE);
1624         //
1625         PrefsRecord record = new PrefsRecord();
1626         //
1627         record.setIndex(0);
1628         SyncManager.readRecordByIndex(db, record);
1629         //
1630         if ((record.getURL() != null) && (!"".equals(record.getURL()))) {
1631             url = record.getURL();
1632             //
1633             url =
1634                 (url.toLowerCase().startsWith("tcp")
1635                  ? "http" + url.substring(url.indexOf(":"))
1636                  : url);
1637         }
1638     } finally {
1639         // Close DB
1640         if (db != -1) {
1641             SyncManager.closeDB(db);
1642         }
1643     }
1644     //
1645     return url;
1646 }
1647 /**
1648  * Method to synchronize the interventions from the HH to a site on the internet.
1649  *
1650  * @param dbName String The name of the database that will be synchronized.
1651  */
1652 private void syncInterventions(String dbName)
1653     throws SyncException, IOException {
1654     Vector syncedRecords = new Vector();
1655     //
1656     String syncURL = readSyncURL("dbPidsPrefs");
1657     //
1658     //=====
1659     //      F I R S T   S Y N C   X M L   D A T A B A S E
1660     //=====
1661     //
1662     // Open our handheld DB
1663     int db =
1664         SyncManager.openDB(
1665             dbName,
1666             0,
1667             SyncManager.OPEN_READ | SyncManager.OPEN_WRITE | SyncManager.OPEN_EXCLUSIVE);
1668
1669     // Find out how many intervention records there are in the database...
1670     int count = SyncManager.getDBRecordCount(db);
1671
1672     XMLInterventionRecord record = null;
1673     //
1674     DOMInterventionURLBuilder urlBuilder = null;
1675     //
1676     // Loop over all intervention records
1677     CharArrayReader reader = null;
1678     //
1679     for (int i = 0; i < count; i++) {
1680         record = new XMLInterventionRecord();
1681         //
1682         record.setIndex(i);
1683         SyncManager.readRecordByIndex(db, record);
1684         //
1685         if (record.getSynced() == 0.0) {
1686             //
1687             // Now use the DOM Parser to build the POST string for the url...
1688             char[] xmlCharArray =
1689                 ((record.getXMLRecord() != null)
1690                  ? record.getXMLRecord().toCharArray()

```

```

1691         : "".toCharArray());
1692     reader = new CharArrayReader(xmlCharArray);
1693     //
1694     try {
1695         urlBuilder = new DOMInterventionURLBuilder();
1696         //
1697         urlBuilder.parse(reader);
1698         //
1699         if (postRequest(urlBuilder.retrievePostString(), syncURL) == 1) {
1700             //
1701             // Now write the intervention back as "synced"...
1702             syncedRecords.addElement(new Integer(record.getId()));
1703         }
1704     } catch (Exception e) {
1705         throw new SyncException(
1706             "PIDS intervention syncing failed for intervention record id "
1707             + record.getId()
1708             + ". This may be the result of an invalid character entered in an
1709             intervention SOAP note that the conduit is unable to process.
1710             Please email beta@pidsware.com if this occurs.");
1711     }
1712 }
1713 }
1714
1715 // Close DB
1716 SyncManager.closeDB(db);
1717
1718 //
1719 //=====
1720 //      NOW WRITE RECORDS BACK
1721 //      FLAGGING IT AS SYNCED
1722 //=====
1723 //
1724 //
1725 db =
1726     SyncManager.openDB(
1727         dbName,
1728         0,
1729         SyncManager.OPEN_READ | SyncManager.OPEN_WRITE | SyncManager.OPEN_EXCLUSIVE);
1730
1731 // Find out how many intervention records there are in the database...
1732 count = SyncManager.getDBRecordCount(db);
1733
1734 record = null;
1735 //
1736 // Loop over all intervention records
1737 Integer id = null;
1738 Enumeration records = syncedRecords.elements();
1739 //
1740 while (records.hasMoreElements()) {
1741     // for (int i = 0; i < count; i++) {
1742     id = (Integer) records.nextElement();
1743     //
1744     record = new XMLInterventionRecord();
1745     //
1746     record.setId(id.intValue());
1747     SyncManager.readRecordById(db, record);
1748     //
1749     // Now write the XML intervention back to the HH flagged as "synced"...
1750     record.setSynced(1.0);
1751     SyncManager.writeRec(db, (Record) record);
1752 }
1753 // Close DB
1754 SyncManager.closeDB(db);
1755 }
1756 }
1757
1758
1759

```



```

1760
1761
1762
1763
1764
1765
1766
1767 package com.pidsware.pids.conduit;
1768
1769 import java.io.*;
1770 import java.util.*;
1771 import java.sql.*;
1772 /**
1773  *
1774  * DOMInterventionURLBuilder.java
1775  *
1776  * #####
1777  *
1778  * PIDS (tm)
1779  * Pharmacy Intervention Documentation System
1780  * PIDS (tm) Conduit for performing Palm HotSync's
1781  * Copyright © 2002 Integrated Software Systems, L.L.C.
1782  *
1783  * #####
1784  *
1785  *
1786  * This class is a non-validating DOM parser which takes in an
1787  * XML formatted intervention, parses it, and dynamically builds
1788  * a URL encoded name/value parameters list that gets sent as a
1789  * POST request to the designated sync URL provided by the user
1790  * on the HH.
1791  *
1792  *
1793  * @version 1.0
1794  * @date 12.15.2001
1795  * @author: Todd Viegut & Brad Tice
1796  */
1797 public class ConduitDebugLog {
1798     private PrintWriter log = null;
1799     static private ConduitDebugLog instance; // The single instance
1800     /**
1801      * DebugLog constructor comment.
1802      */
1803     private ConduitDebugLog() {
1804         init();
1805     }
1806     public static synchronized ConduitDebugLog getInstance() {
1807         if (instance == null) {
1808             instance = new ConduitDebugLog();
1809         }
1810         return instance;
1811     }
1812     /**
1813      * Loads properties and initializes the instance with its values.
1814      */
1815     private void init() {
1816         InputStream is = getClass().getResourceAsStream("PIDScond.properties");
1817         Properties dbProps = new Properties();
1818         try {
1819             dbProps.load(is);
1820             String logFile = dbProps.getProperty("logfile", "PIDScond.log");
1821             try {
1822                 log = new PrintWriter(new FileWriter(logFile, true), true);
1823             } catch (IOException e) {
1824                 System.err.println("Can't open the log file: " + logFile);
1825                 log = new PrintWriter(System.err);
1826             }
1827         } catch (Exception e) {
1828             System.err.println("Can't open the log file: " + "PIDSCond.log" + " Using
1829             default logging to standard err.");
1830             log = new PrintWriter(System.err);

```

```
1830     }
1831
1832 }
1833 /**
1834  * Writes a message to the log file.
1835  */
1836 public synchronized void log(Object obj, String msg) {
1837     log.println(obj.getClass().getName() + " [" + new java.util.Date() + "]: " + msg);
1838 }
1839 /**
1840  * Writes a message to the log file.
1841  */
1842 public synchronized void log(Object obj, String msg, Throwable exception) {
1843     log.println(obj.getClass().getName() + " [" + new java.util.Date() + "]: " + msg);
1844     exception.printStackTrace(log);
1845 }
1846 /**
1847  * Writes a message to the log file.
1848  */
1849 public synchronized void log(String msg) {
1850     log.println("[" + new java.util.Date() + "]: " + msg);
1851 }
1852 /**
1853  * Writes a message with an Exception to the log file.
1854  */
1855 public synchronized void log(String msg, Throwable exception) {
1856     log.println("[" + new java.util.Date() + "]: " + msg);
1857     exception.printStackTrace(log);
1858 }
1859 /**
1860  * Starts the application.
1861  * @param args an array of command-line arguments
1862  */
1863 public static void main(java.lang.String[] args) {
1864     // Insert code to start the application here
1865 }
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877 package com.pidsware.pids.conduit;
1878
1879 import palm.conduit.*;
1880 import java.io.*;
1881 import java.net.URLEncoder;
1882 /**
1883  *
1884  * PrefsRecord.java
1885  *
1886  #####
1887  *
1888  * PIDS (tm)
1889  * Pharmacy Intervention Documentation System
1890  * PIDS (tm) Conduit for performing Palm HotSync's
1891  * Copyright © 2002 Integrated Software Systems, L.L.C.
1892  *
1893  #####
1894  *
1895  *
1896  * Subclassed record object for the a preference database record on
1897  * the HM.
1898  *
1899  *
1900  * @version 1.0
```

```
1901  * @date    02.10.2002
1902  * @author:  Todd Viegut & Brad Tice
1903  */
1904  public class PrefsRecord extends AbstractRecord {
1905      private double prefsKey;
1906      private double enableSpeedEntry;
1907      private String url = null;
1908      private String locationID = null;
1909      private String rotationID = null;
1910      private String userID = null;
1911      public double getEnableSpeedEntry() {
1912          return this.enableSpeedEntry;
1913      }
1914      public String getLocationID() {
1915          return this.locationID;
1916      }
1917      public double getPrefsKey() {
1918          return this.prefsKey;
1919      }
1920      public String getRotationID() {
1921          return this.rotationID;
1922      }
1923      public String getURL() {
1924          return this.url;
1925      }
1926      public String getUserID() {
1927          return this.userID;
1928      }
1929      String nullIfEmpty(String s) {
1930          if ("".equals(s)) {
1931              return null;
1932          } else {
1933              return s;
1934          }
1935      }
1936      public void readData(DataInputStream in) throws IOException {
1937          //
1938          //=====
1939          //          P R E F E R E N C E S
1940          //=====
1941          //
1942          // PIDS intervention key
1943          this.prefsKey = in.readDouble();
1944          // Lab Values
1945          this.enableSpeedEntry = in.readDouble();
1946          // Wireless URL
1947          this.url = readCString(in);
1948          // Location ID
1949          this.locationID = readCString(in);
1950          // Rotation Date
1951          this.rotationID = readCString(in);
1952          // User ID
1953          this.userID = readCString(in);
1954      }
1955      public void setLocationID(String value) {
1956          this.locationID = nullIfEmpty(value);
1957      }
1958      public void setPrefsKey(double value) {
1959          this.prefsKey = value;
1960      }
1961      public void setRotationID(String value) {
1962          this.rotationID = nullIfEmpty(value);
1963      }
1964      public void setURL(String url) {
1965          this.url = nullIfEmpty(url);
1966      }
1967      public void setUserID(String userID) {
1968          this.userID = nullIfEmpty(userID);
1969      }
1970      public void setEnableSpeedEntry(double value) {
1971          this.enableSpeedEntry = value;
1972      }
```

```

1972 }
1973 public void setUserID(String value) {
1974     this.userID = nullIfEmpty(value);
1975 }
1976 static String stringWithoutCarriageReturns(String aString) {
1977     StringBuffer sb = new StringBuffer();
1978     int i, c;
1979     char ch;
1980
1981     for (i = 0, c = aString.length(); i < c; i++) {
1982         ch = aString.charAt(i);
1983         if (ch == '\r' && (i + 1) < c && aString.charAt(i + 1) == '\n')
1984             continue;
1985         else
1986             sb.append(ch);
1987     }
1988     return sb.toString();
1989 }
1990 public String toFormattedString() {
1991     return (
1992         "Intervention record: {\r\n"
1993         + "    key: "
1994         + getPrefsKey()
1995         + "\r\n"
1996         + "    enable speed entry: "
1997         + getEnableSpeedEntry()
1998         + "\r\n"
1999         + "    url: "
2000         + getURL()
2001         + "\r\n"
2002         + "    location ID: "
2003         + getLocationID()
2004         + "\r\n"
2005         + "    rotation ID: "
2006         + getRotationID()
2007         + "\r\n"
2008         + "    user ID: "
2009         + getUserID()
2010         + "\r\n"
2011         + "}" + "\r\n"
2012         + super.toFormattedString());
2013 }
2014 public String toString() {
2015     return "prefs record:  <enable speed entry="
2016         + getEnableSpeedEntry()
2017         + ">  <url="
2018         + getURL()
2019         + ">  <rotation id="
2020         + getRotationID()
2021         + ">  <location id="
2022         + getLocationID()
2023         + ">  <user id="
2024         + getUserID()
2025         + ">  "
2026         + super.toString();
2027 }
2028 public void writeData(DataOutputStream out) throws IOException {
2029
2030     // PIDS intervention key
2031     out.writeDouble(this.prefsKey);
2032
2033     // Synced
2034     out.writeDouble(this.enableSpeedEntry);
2035
2036     // URL
2037     if (this.url != null) {
2038         out.write(this.url.getBytes());
2039         out.write(0);
2040     }
2041
2042     // Location ID

```

```

2043     if (this.locationID != null) {
2044         out.write(this.locationID.getBytes());
2045         out.write(0);
2046     }
2047
2048     // Rotation ID
2049     if (this.rotationID != null) {
2050         out.write(this.rotationID.getBytes());
2051         out.write(0);
2052     }
2053
2054     // User ID
2055     if (this.userID != null) {
2056         out.write(this.userID.getBytes());
2057         out.write(0);
2058     }
2059 }
2060 }
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071 package com.pidsware.pids.conduit;
2072
2073 import java.io.*;
2074 import palm.conduit.*;
2075 /**
2076  *
2077  * XMLInterventionRecord.java
2078  *
2079  * #####
2080  *
2081  * PIDS (tm)
2082  * Pharmacy Intervention Documentation System
2083  * PIDS (tm) Conduit for performing Palm HotSync®s
2084  * Copyright © 2002 Integrated Software Systems, L.L.C.
2085  *
2086  * #####
2087  *
2088  *
2089  * Subclassed record object for the an intervention stored as XML on
2090  * the HH.
2091  *
2092  *
2093  * @version 1.0
2094  * @date 02.04.2002
2095  * @author: Todd Viegut & Brad Tice
2096  */
2097 public class XMLInterventionRecord extends AbstractRecord {
2098     private double interventionKey;
2099     private double synced;
2100     private String xmlRecord = null;
2101     public double getInterventionKey() {
2102         return this.interventionKey;
2103     }
2104     public double getSynced() {
2105         return this.synced;
2106     }
2107     /** Converts and copies dbPidsXML members into a DataOutputStream in preparation
2108      * for writing to the hand-held device.
2109      * The format of the DataOutputStream specific to the record body byte
2110      * layout of the memo records stored on the hand-held device for the PIDS
2111      * application.
2112      * @param out The DataOutputStream to fill with XML record body information
2113      */

```

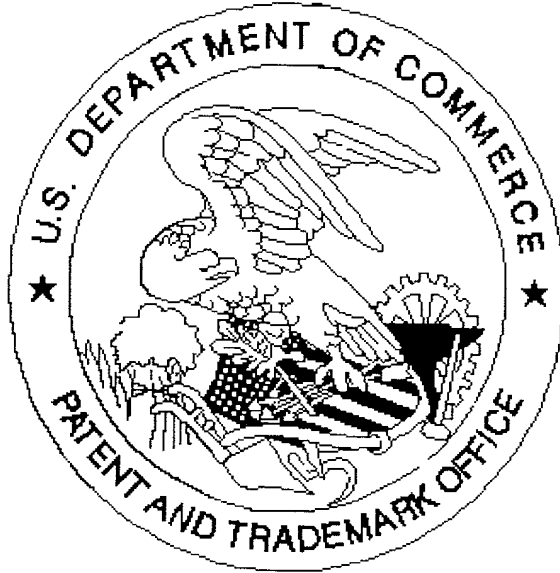
```

2114
2115 /** Returns the XML record.
2116 */
2117 public String getXMLRecord() {
2118     return this.xmlRecord;
2119 }
2120 String nullIfEmpty(String s) {
2121     if ("".equals(s)) {
2122         return null;
2123     } else {
2124         return s;
2125     }
2126 }
2127 public void readData(DataInputStream in) throws IOException {
2128     //
2129     //=====
2130     //      I N T E R V E N T I O N   H E A D E R
2131     //=====
2132     //
2133
2134     // PIDS intervention key
2135     this.interventionKey = in.readDouble();
2136
2137     // Synced flag
2138     this.synced = in.readDouble();
2139
2140     //
2141     //=====
2142     //      X M L   I N T E R V E N T I O N   R E C O R D
2143     //=====
2144     //
2145
2146     // PIDS Intervention record as XML
2147     this.xmlRecord = readCString(in);
2148 }
2149 public void setInterventionKey(double value) {
2150     this.interventionKey = value;
2151 }
2152 public void setSynced(double value) {
2153     this.synced = value;
2154 }
2155 public void setXMLRecord(String xmlRecord) {
2156     this.xmlRecord = xmlRecord;
2157 }
2158 static String stringWithoutCarriageReturns(String aString) {
2159     StringBuffer sb = new StringBuffer();
2160     int i, c;
2161     char ch;
2162
2163     for (i = 0, c = aString.length(); i < c; i++) {
2164         ch = aString.charAt(i);
2165         if (ch == '\r' && (i + 1) < c && aString.charAt(i + 1) == '\n')
2166             continue;
2167         else
2168             sb.append(ch);
2169     }
2170     return sb.toString();
2171 }
2172 public String toFormattedString() {
2173     StringBuffer buffer = new StringBuffer();
2174
2175     buffer.append("\n\n=====").append("\n\n")
2176         .append("INTERVENTION").append("\n")
2177         .append("-----").append("\n\n")
2178         .append("Intervention Key:  ").append(getInterventionKey()).append("\n")
2179         .append("Synced Flag:      ").append(getSynced()).append("\n\n")
2180         .append("XML Record:").append("\n")
2181         .append(getXMLRecord()).append("\n\n")
2182         .append(super.toFormattedString()).append("\n")
2183         .append("-----").append("\n");

```

```
2183     return buffer.toString();
2184 }
2185 public String toString() {
2186     StringBuffer buffer = new StringBuffer();
2187     buffer.append("XML Intervention Record:").append("\n")
2188         .append("key:      ").append(getInterventionKey()).append("\n")
2189         .append("synced:   ").append(getSynced()).append("\n")
2190         .append("xml:      ").append(getXMLRecord()).append("\n")
2191         .append(super.toString()).append("\n");
2192     return buffer.toString();
2193 }
2194 public void writeData(DataOutputStream out) throws IOException {
2195     // XML record
2196     /*
2197     if (this.xmlRecord != null) {
2198         out.write(this.xmlRecord.getBytes());
2199         out.write(0);
2200     }
2201     */
2202     // PIDS intervention key
2203     out.writeDouble(this.interventionKey);
2204     // Synced
2205     out.writeDouble(this.synced);
2206     // Rotation ID
2207     if (this.xmlRecord != null) {
2208         out.write(this.xmlRecord.getBytes());
2209         out.write(0);
2210     }
2211     //writeCString(out, stringWithoutCarriageReturns(this.xmlRecord));
2212 }
2213 }
2214 }
2215 }
2216 }
2217 }
```

United States Patent & Trademark Office
Office of Initial Patent Examination -- Scanning Division



Application deficiencies found during scanning:

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

** Scanned copy is best available. There are 60 page of specific
- cation. Rest of the pages are Appendix.*